HOCHSCHULE FURTWANGEN UNIVERSITY HFU

Fakultät Mechanical and Medical Engineering

Bachelor-Thesis

Motorsteuerung für Asynchronmotoren in Gleichstromnetzwerken

von

Dominik Rajsp

Elektronik und Technische Informatik

Prüfer

1. Prüfer: Prof. Dr.-Ing. Robert Hönl, HFU

2. Prüfer: Dr. Volker Lange, HFU

Kurzfassung/Abstract

Thema:	**Motorsteuerung für Asynchronmotoren in Gleichstromnetzwerken**
Topic:	**Development of an Induction Motor Drive for Operation in DC Networks**
Verfasser/Author:	Dominik Rajsp
1. Prüfer/Examiner:	Prof. Dr.-Ing. Robert Hönl, HFU
2. Prüfer/Examiner:	Dr. Volker Lange, HFU
Semester:	Sommersemester 2018/ summer semester 2018

In dieser Arbeit soll eine Motorsteuerung für einen 3-Phasen-Asynchronmotor mit Kurzschlussläufer realisiert werden. Dabei soll der Motor in einem Gleichstromnetzwerk mit einer Spannung von 325V betrieben werden. Drehzahl und Drehrichtung des Motors sollen variabel sein. Der Motor wird dabei bei konstantem Lastdrehmoment betrieben. Des Weiteren soll die Steuerung eine Drehzahl des Motors von bis zu $25000 min^{-1}$ ermöglichen.
Die Arbeit behandelt dabei einführend die Funktionsweise des Asynchronmotors und leitet ein mathematisches Modell ab, das anschließend für die Erstellung des Steueralgorithmus verwendet wird. Bei der Realisierung der Schaltung wird u.a. die Problematik mit parasitären Induktivitäten und der Umgang mit motorinduzierten Spannungsspitzen behandelt. Realisiert wird die Motorsteuerung als Umrichterschaltung. Dabei finden ausschließlich N-Kanal-MOSFETs Verwendung. Für die Ansteuerung der MOSFETs kommen Gate-Treiber-ICs mit Bootstrap-Kondensatoren zum Einsatz. Angesteuert wird der Motor über einen STM32-Board. Der implementierte Steueralgorithmus realisiert dabei eine Grundfrequenztaktung. Über diesen Aufbau der Motorsteuerung können die Anforderungen erfüllt werden.

This thesis describes the development of an motor drive for an induction motor with squirrel-cage. The induction motor operates in a DC network with a voltage of 325V. The motors rotation speed and the direction of rotation should be variable. The motor drive should be able to operate the motor at a rotation speed up to 25000 rpm.
At the beginning, the induction motor is described. Then a mathematical model is developed. This model is used to create the control algorithm. For the motor drive, MOSFET-bridges are used to switch the motor phases between 325V and GND. Only n-channel MOSFETs are used. To drive the MOSFETS, gate driver ICs with bootstrap capacitors were used in the circuit. The typical problems with parasitic inductances and motor induced voltage peaks are described. Then solutions for these problems are discussed. The algorithm is implemented on an STM32 board, wich will send the control signals to the gate driver ICs. With this structure of the motor drive, the requirements could be fulfilled.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne unzulässige fremde Hilfe angefertigt habe.

Alle verwendeten Quellen (Literatur, Internet) sind im Literaturverzeichnis vollständig zitiert.

Furtwangen, den 21. Oktober 2018

Dominik Rajsp

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

1 Einführung

Im Zuge dieser Thesis soll eine Steuerung für einen 3-Phasen-Asynchronmotor mit Kurzschlussläufer (ASM) entwickelt werden. Dabei wird für die Regelung ein Betrieb des ASM in einem Gleichstromnetzwerk zugrunde gelegt. Für die Regelung gelten folgende Rahmenbedingungen:

- die Regelung soll eine maximale Drehzahl des Motors von bis zu 25000 min^{-1} ($\approx 416Hz$) ermöglichen
- die Gleichspannung für den Motor soll 325V betragen
- der Motor soll bei konstantem Drehmoment betrieben werden (stationärer Betrieb)
- die Drehzahl des Motors soll variabel sein
- die Drehrichtung des Motors soll einstellbar sein

Um die Regelung realisieren zu können, muss zuerst die Funktionweise eines ASM und dessen mathematische Beschreibung erarbeitet werden. Dies geschieht in Kapitel 2. Anschließend wird das geplante Regelverfahren vorgestellt und daraus der Regelalgorithmus für den Mikroprozessor abgeleitet (Kapitel 3). Dabei müssen Annahmen bezüglich der Motorparameter und deren Temperatur- und Frequenzabhängigkeit getroffen werden, die auf ihre Zulässigkeit (für den in dieser Arbeit verwendeten Motor und die gesetzten Rahmenbedingungen) hin diskutiert werden. Die Realisierung der Hard- und Software wird in Kapitel 4 und 5 beschrieben. Dabei wird u.a. auf die Reduzierung von parasitären Induktivitäten und die Timing-Problematik bei H-Brücken eingegangen. Abschließend wird das Verhalten der realisierten Regelung in Bezug auf die oben aufgeführten Rahmenbedingungen diskutiert und in einem Ausblick mögliche Ansatzpunkte für eine Verbesserung der Regelung beschrieben.

2 Theoretische Grundlagen

In diesem Kapitel wird einführend der Aufbau und die Funktionsweise eines ASM beschrieben und anschließend ein geeignetes mathematisches Modell für den Motor abgeleitet. Dieses Modell wird dann für die Entwicklung des Regelalgorithmus (Kapitel 3) herangezogen.

2.1 Aufbau und Funktionsweise eines ASM

Asynchronmotoren sind in verschiedenen Bauformen erhältlich. Im Rahmen dieser Thesis wird ein ASM verwendet. Die Beschreibung des Aufbaus und der Funktionsweise beschränkt sich daher auf diese Bauform. Für eine detailliertere Beschreibung von Aufbau und Funktionsweise des ASM sei an dieser Stelle auf [14, Kapitel 5.2 ff] verwiesen.

2.1.1 Aufbau eines ASM

Ein ASM besteht im Wesentlichen aus einem Stator, der die Spulen beinhaltet, und einem Rotor (auch Läufer genannt), der mit der Motorwelle verbunden ist. Rotor, Welle und Wellenlagerung sind bei der ASM die einzigen mechanisch rotierenden Komponenten. Die Anzahl der Spulen und der genaue Aufbau des Rotors variieren in der Praxis.
In Abbildung 2.1a ist der Querschnitt eines stark vereinfachten ASM abgebildet. Der Rotor ist vom Stator umgeben und rotiert (bei korrekter Ansteuerung der Spulen) um die Rotationsachse mit der Winkelgeschwindigkeit $\omega(t)$. Für die Funktionsweise des ASM ist die Form des Magnetfeldes im Luftspalt $\vec{B}_{LS}$ von zentraler Bedeutung. $\vec{B}_{LS}$ zeigt über die gesamte Breite des Luftspaltes nahezu radial nach außen bzw. nach innen (verursacht durch die geringe Breite des Luftspalts und durch die Brechungsgesetze für Magnetfeldlinien) [14]. Wird $\vec{B}_{LS}$ bei der Speißung einer einzelnen Spule mit einem konstanten Strom betrachtet und $|\vec{B}_{LS}|$ über den räumlichen Winkel ε aufgetragen, so gleicht die Idealform des Betrages einem Sinusverlauf (durch den Sinusverlauf wird u.a. eine Glättung des Drehmoments erreicht) [14]. In der Realität wird über eine geeignete Aufteilung der Spule auf mehrere Nuten um den Rotor herum, der gewünschte Sinusverlauf von $|\vec{B}_{LS}|$ angenähert [14]. In Abbildung 2.1b ist der prinzipielle Aufbau eines Kurzschlussläufers abgebildet. Dieser Aufbau zeigt, dass die elektrisch leitfähigen Läuferstäbe mit den elektrisch leitfähigen Abschlussringen kurzgeschlossen werden. Dadurch bilden je zwei Läuferstäbe und zwei Teilstücke der Abschlussringe eine geschlossene Leiterschleife (Näheres dazu im Abschnitt 2.1.2). Der Läufer wird von einem Eisenkern ausgefüllt. Durch den Querschnitt des Eisenkerns verläuft bei idealem Sinusverlauf von $|\vec{B}_{LS}|$ ein nahezu homogenes Magnetfeld, dessen Richtung zum Maximum von $|\vec{B}_{LS}|$ zeigt [14].

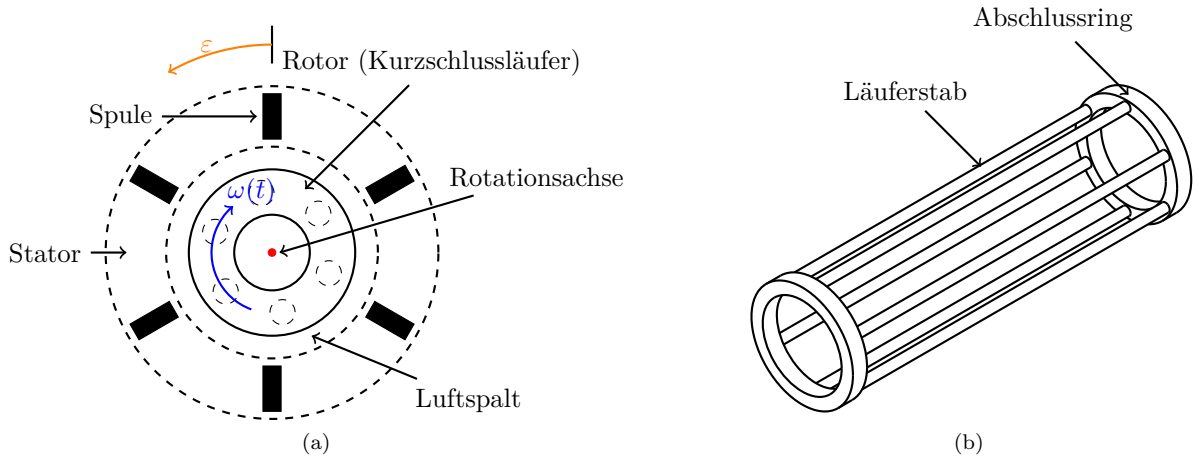


Abbildung 2.1: (a) Querschnitt durch einen ASM (vereinfachter Aufbau) mit $0 \leq \varepsilon < 2\pi$, (b) prinzipieller Aufbau eines Kurzschlussläufers (Eisenkern nicht eingezeichnet) (Quelle der Originalgrafik: [14], Beschriftung hinzugefügt)

2.1.2 Funktionsweise eines ASM

Die Funktionsweise eines ASM basiert zum einen auf dem Induktionsgesetz (Gl. 2.1) und zum anderen auf der Lorentz-Kraft (Gl. 2.2) [9]. Durch die Spulen im Stator wird ein Magnetfeld im Rotor erzeugt, das in den geschlossenen Leiterschleifen des Kurzschlussläufers Spannungen induziert. Diese Spannungen erzeugen Ströme in den Leiterschleifen, die nach Gl. 2.2 mit dem resultierenden Magnetfeld im Luftspalt (Abschnitt 2.1.1) eine Lorentz-Kraft für jede Leiterschleife erzeugen. Die aus den einzelnen Lorentz-Kräften resultierende Gesamtkraft bewirkt dann (nach Betrag und räumlicher Ausrichtung) eine Änderung der Winkelgeschwindigkeit $\omega(t)$ (Abb. 2.1a).

$$U_{ind} = \oint_{\partial A} \vec{E} d\vec{s} = -\int_A \frac{\partial \vec{B}}{\partial t} d\vec{A} \tag{2.1}$$

$$\vec{F}_L = I \int d\vec{l} \times \vec{B} \tag{2.2}$$

- $\vec{E}$: elektrische Feldstärke
- $\vec{B}$: magnetische Flussdichte
- A: Fläche, in der die zeitliche Änderung von $\vec{B}$ betrachtet wird (entspricht hier der Fläche, die von der Leiterschleife umschlossen wird)
- ∂A: Rand der Fläche A (entspricht hier der Leiterschleife)
- I: Stromstärke
- $d\vec{l}$: infinitesimal kleines Leiterstück, das sich im Magnetfeld befindet

Gl. 2.1 zeigt, dass nur dann eine Spannung induziert wird, wenn sich das Magentfeld zeitlich ändert. Da der Läufer mit einer Winkelgeschwindigkeit $\omega(t) \neq 0$ rotieren kann, muss sich das Magnetfeld **relativ** zum Läufer zeitlich ändern. Das Magnetfeld muss also auch mit einer Winkelgeschwindigkeit $\omega_B(t) \neq \omega(t)$ rotieren. Zur Erzeugung des rotierenden Magnetfeldes werden noch zwei weitere Spulen in die Betrachtungen mit einbezogen. Die beiden Spulen werden ebenfalls so auf mehrere Nuten des Stators verteilt, dass sie bei Speißung mit konstantem Strom ebenfalls ein Magnetfeld mit sinusförmigem Verlauf von $|\vec{B}_{LS}|$ erzeugen. Die drei Spulen werden nun räumlich um 120° versetzt um den Läufer angeordnet. Werden die Spulen mit um 120° gegeneinander phasenverschobenen sinusförmigen Strömen gespeißt, ensteht ein Gesamtmagnetfeld $\vec{B}_{res}$, das einen rotierenden sinusförmigen Verlauf von $|\vec{B}_{res,LS}|$ aufweist ($\vec{B}_{res,LS}$ = resultierendes Magnetfeld im Lustspalt) [14]. Ziel der Steuerung wird es sein, eine Sinusform der Ströme zu erzeugen bzw. möglichst gut anzunähern.

2.2 Raumzeigerdarstellung

Eine gängige mathematische Beschreibung der Größen des Motors verwendet sogenannte Raumzeiger (RMZ). Bei einem RMZ handelt es sich um eine komplexe Zahl nach Gl. 2.3.

$$z(t) = v(t) + jw(t) = r(t) \cdot e^{j\omega(t)} \tag{2.3}$$
$$v, w, r, \omega, t \in \mathbb{R}$$
$$t : \text{Zeit}$$

Der RMZ ist das Ergebnis einer Transformation der realen Größen des Motors in die komplexe Zahlenebene. Die Transformationsvorschrift wird nach [14] wie folgt angesetzt:

$$\underline{X}(t) = \frac{2}{3}\left(X_a(t) + e^{j\frac{2}{3}\pi} \cdot X_b(t) + e^{-j\frac{2}{3}\pi} \cdot X_c(t)\right) \cdot e^{-j\alpha(t)} \tag{2.4}$$

Unterstrichene Buchstaben kennzeichnen ab sofort einen RMZ. In Abbildung 2.2b ist Gl. 2.4 grafisch dargestellt. In Abbildung 2.2a sind die drei um 120° (bzw. $\frac{2}{3}\pi$) räumlich versetzten Spulen mit ihren Spulenachsen dargestellt. Dabei sind die Spulen konzentriert gezeichnet. Dies dient der einfacheren Darstellung, da die einzelnen Spulen in der Realität auf mehrere Nuten des Stators aufgeteilt sind (siehe Abschnitt 2.1.1). Die Information über die räumliche Anordnung der Spulen wird über die komplexen Faktoren $e^{j\frac{2}{3}\pi}$ bzw. $e^{-j\frac{2}{3}\pi}$ (Gl. 2.4) in die komplexe Zahlenebene übertragen. Der Winkel $\alpha(t)$ beschreibt

eine Drehung der drei roten Zeiger in Bezug auf die reelle Achse. Der Vorteil eines RMZ besteht u.a. darin, dass man mit ihm drei sich zeitlich ändernde, um 120° räumlich versetzte Größen mit einer komplexen Größe beschreiben kann [14]. Die Rücktransformation von einem RMZ auf die realen Größen erfolgt unter der Annahme von $\alpha(t) = 0$ und $X_a(t) + X_b(t) + X_c(t) = 0$ nach [14] wie folgt:

$$X_a(t) = \Re e\{\underline{X}(t)\} \tag{2.5}$$

$$X_b(t) = \frac{1}{2} \cdot \left(\sqrt{3} \cdot \Im m\{\underline{X}(t)\} - \Re e\{\underline{X}(t)\}\right) \tag{2.6}$$

$$X_c(t) = \frac{1}{2} \cdot \left(-\sqrt{3} \cdot \Im m\{\underline{X}(t)\} - \Re e\{\underline{X}(t)\}\right) = -X_a(t) - X_b(t) \tag{2.7}$$

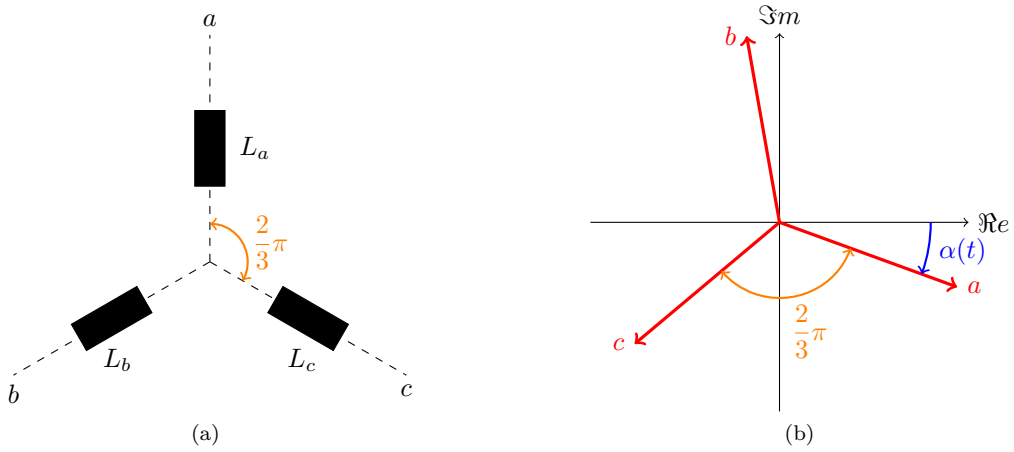


Abbildung 2.2: (a) räumliche Anordnung der zusammengefassten Strangspulen des ASM, (b) Transformation der drei Spulenachsen a, b und c in die komplexe Zahlenebene

Für $\alpha(t) = 0$ wird die Achse a auf die reelle Achse der komplexen Zahlenebene abgebildet. Da die Spulenachsen raumfest sind, sind damit auch die reelle und imaginäre Achse raumfest. Dieses Koordinatensystem wird als S-System (Stator-System) definiert [14] und die reelle Achse als x-Achse, die imaginäre Achse als y-Achse bezeichnet. Es können nun weitere Koordinatensysteme definiert werden, die mit beliebiger Kreisfrequenz relativ zum S-System rotieren. Die richtige Wahl dieser Kreisfrequenz und die Betrachtung der komplexen Größen in diesem rotierenden Koordinatensystem kann zu einer erheblichen Vereinfachung der Formeln führen (vergleichbar mit dem Übergang von kartesischen in Polarkoordinaten bei Rotationsproblemen). In den folgenden Kapiteln wird jedoch nur das S-System benötigt. Alle RMZ werden, **wenn nicht explizit anders angegeben**, im S-System betrachtet.

2.3 Mathematische Modellierung des Motors

Für die mathematische Beschreibung des ASM sind u.a. die Zeit-, Frequenz- und Temperaturabhängigkeiten der Motorparameter (z.B. der Motorinduktivitäten) von zentraler Bedeutung. Je genauer diese Abhängigkeiten in das Motormodell mit einbezogen werden, desto genauer (aber auch komplexer) wird das mathematische Modell. Für die in Kapitel 1 aufgeführten Rahmenbedingungen kann eines der einfachsten Motormodelle angesetzt werden. In Tabelle 2.1 sind für dieses Motormodell die getroffenen Annahmen aufgelistet. Des Weiteren können aus der Realisierung des Motors zusätzliche Zusammenhänge abgeleitet werden. Im Allgemeinen ist ein ASM symmetrisch aufgebaut (so auch der hier verwendete Motor), d.h. die elektrischen Widerstände und Induktivitäten der einzelnen Phasen sind annähernd gleich. Für das hier verwendete Motormodell können die Widerstände und Induktivitäten jeweils als identisch angenommen werden, da die Unterschiede sehr gering ausfallen.

$$R_a = R_b = R_c = R \tag{2.8}$$

$$L_a = L_b = L_c = L \tag{2.9}$$

Damit können für eine Phase folgende mathematische Gleichungen angesetzt werden (hier für Phase a):

Formelzeichen	**Beschreibung**	**für das Motormodell angesetztes Verhalten**
R_a, R_b, R_c	elektrischer Widerstand einer Motorphase	keine Zeit-, Frequenz- und Temperaturabhängigkeit
L_a, L_b, L_c	Induktivität einer Motorphase	keine Zeit-, Frequenz- und Temperaturabhängigkeit
J	Massenträgheitsmoment des Rotors	keine Zeitabhängigkeit
-	magnetische Sättigungseffekte	werden vernachlässigt
$\lvert\vec{B}_{LS,a}\rvert$, $\lvert\vec{B}_{LS,b}\rvert$, $\lvert\vec{B}_{LS,c}\rvert$	Betrag der magnetische Flussdichte der einzelnen Spulen im Luftspalt	haben den in Unterkapitel 2.1.1 beschriebenen idealen Sinusverlauf
-	Eisenverluste	werden vernachlässigt
-	Stromverdrängung	wird vernachlässigt
-	Reibungsverluste in der Lagerung des Rotors	werden vernachlässigt
-	Verluste durch den im Motor integrierten Lüfter	werden vernachlässigt

Tabelle 2.1: Annahmen für das Motormodell

$$U_a(t) = R \cdot I_a(t) + \frac{d\Psi_a}{dt}(t) \tag{2.10}$$

$$\Psi_a(t) = L \cdot I_a(t) + M \cdot I_{R,a}(t) \tag{2.11}$$

Gl. 2.10 stellt die Maschengleichung einer Phase dar. Gl. 2.11 beschreibt den magnetischen Fluss in der Motorspule a als eine Linearkombination aus Spulenstrom I_a und Rotorstrom $I_{R,a}$. L ist die Eigeninduktivität der Motorspule und M ist die maximale Gegeninduktivität zwischen den Motorspulen und dem Rotor. M beschreibt die Stärke der magnetischen Rückkopplungen des Rotors auf den Stator. Gl. 2.11 wird auch als Flussverkettung bezeichnet [14]. Für Phase b und c können analog zur Phase a je eine Maschengleichung und eine Flussverkettungsgleichung aufgestellt werden. Mit Gl. 2.4 können die drei Maschengleichungen und die drei Flussverkettungsgleichungen in RMZ transformiert werden ($\alpha(t) = 0$). Damit ergeben sich die folgenden beiden Gleichungen:

$$\underline{U}_S(t) = R \cdot \underline{I}_S(t) + \frac{d\underline{\Psi}_S}{dt}(t) \tag{2.12}$$

$$\underline{\Psi}_S(t) = L \cdot \underline{I}_S(t) + M \cdot \underline{I}_R(t) \tag{2.13}$$

Der Index S kennzeichnet Statorgrößen, der Index R Rotorgrößen. Um die Gleichung für das Drehmoment im S-System zu erhalten, wird zuerst die Erzeugung des Drehmoments für eine Spule hergeleitet. Dies wird z.B. in [3, Kapitel 4.7] beschrieben. Als Basis dienen die Überlegungen aus Kapitel 2.1. Aus den drei Gleichungen für die einzelnen Spulen kann dann (nach geeigneter Umformung) mit Gl. 2.4 die folgende Drehmomentgleichung abgeleitet werden:

$$M(t) = \frac{3}{2} \cdot p \cdot \Im m\left(\underline{\Psi}_S^*(t) \cdot \underline{I}_S(t)\right) \tag{2.14}$$

- p: Polpaarzahl (Anzahl der Spulen-Triple die gegeneinander um 120° räumlich versetzt angeordnet sind. In Abbildung 2.2a ist z.B. nur ein Triple vorhanden, daher wäre hier p=1)
- $\underline{X}^*$: komplex konjugierter RMZ

Für die Winkelgeschwindigkeit des Rotors kann folgende Differentialgleichung angesetzt werden:

$$J \cdot \frac{d\omega_R}{dt}(t) = M(t) - M_{Last}(t) \tag{2.15}$$

- J: Massenträgheitsmoment (beinhaltet das Massenträgheitsmoment des Rotors **und** der Last)
- M_{Last}: Lastdrehmoment

Die Gleichungen 2.12 bis 2.15 stellen das mathematische Modell dar, auf dem der Regelalgorithmus aufbaut.

3 Der Regelalgorithmus

Für die Motorsteuerung soll das Direct Torque Control (DTC)-Regelverfahren nach I. Takahashi implementiert werden. Die Wahl fiel auf dieses Regelverfahren, da es zum einen auch dynamische Belastungen des Motors ausregeln kann und zum anderen (bei ausreichender Genauigkeit des Motormodells) ohne zusätzliche Sensoren am Motor auskommt [15]. Außerdem ist der Berechnungsaufwand im Vergleich zu anderen Regelverfahren gering [15]. In den folgenden Unterkapiteln wird das Grundprinzip einer DTC-Regelung erläutert und anschließend der zu implementierende Regelalgorithmus abgeleitet.

3.1 Das DTC-Regelverfahren

Für dieses Verfahren wird ein Umrichter benötigt. Ein möglicher Aufbau eines solchen Umrichters ist in Abb. 3.1 abgebildet. Die sechs N-Kanal-MOSFETs ziehen die Motorphasen entweder auf GND oder auf die Versorgungsgleichspannung U_{CC}. Mit den Überlegungen aus Unterkapitel 2.1 können insgesamt acht zulässige Schaltkombinationen für die MOSFETs abgeleitet werden. Für diese Schaltkombinationen können dann die Phasenspannungen $U_a(t)$, $U_b(t)$ und $U_c(t)$ (siehe Abb. 3.1) ermittelt werden. Mit Gl. 2.4 kann dann aus den Phasenspannungen für jede Schaltkombination ein RMZ berechnet werden. In Tabelle 3.1 sind die zulässigen Schaltkombinationen und die zugehörigen Spannungs-RMZ aufgelistet. Um nun die gewünschte Winkelgeschwindigkeit des Rotors einzuregeln, wird beim DTC-Verfahren der Betrag von $\underline{\Psi}_S(t)$ innerhalb eines Toleranzbandes um einen Sollwert gehalten und zusätzlich das Drehmoment an einen Drehmoment-Sollwert angepasst [15].

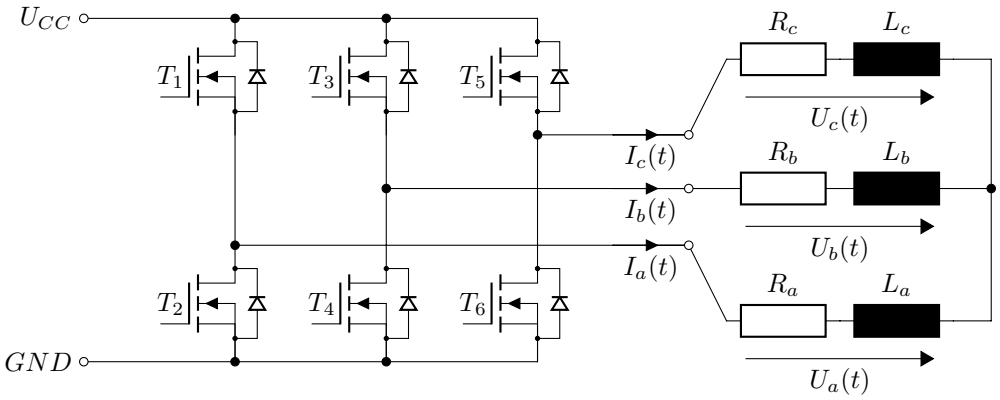


Abbildung 3.1: prinzipieller Aufbau eines Umrichters mit angeschlossenem Motor als Sternschaltung

T_1	T_2	T_3	T_4	T_5	T_6	U_a	U_b	U_c	Spannungsraumzeiger
ein	aus	aus	ein	aus	ein	$\frac{2}{3}U_{CC}$	$-\frac{1}{3}U_{CC}$	$-\frac{1}{3}U_{CC}$	$\underline{U}_1 = \frac{2}{3}U_{CC} \cdot e^{j0}$
ein	aus	ein	aus	aus	ein	$\frac{1}{3}U_{CC}$	$\frac{1}{3}U_{CC}$	$-\frac{2}{3}U_{CC}$	$\underline{U}_2 = \frac{2}{3}U_{CC} \cdot e^{j\frac{\pi}{3}}$
aus	ein	ein	aus	aus	ein	$-\frac{1}{3}U_{CC}$	$\frac{2}{3}U_{CC}$	$-\frac{1}{3}U_{CC}$	$\underline{U}_3 = \frac{2}{3}U_{CC} \cdot e^{j\frac{2}{3}\pi}$
aus	ein	ein	aus	ein	aus	$-\frac{2}{3}U_{CC}$	$\frac{1}{3}U_{CC}$	$\frac{1}{3}U_{CC}$	$\underline{U}_4 = \frac{2}{3}U_{CC} \cdot e^{j\pi}$
aus	ein	aus	ein	ein	aus	$-\frac{1}{3}U_{CC}$	$-\frac{1}{3}U_{CC}$	$\frac{2}{3}U_{CC}$	$\underline{U}_5 = \frac{2}{3}U_{CC} \cdot e^{j\frac{4}{3}\pi}$
ein	aus	aus	ein	ein	aus	$\frac{1}{3}U_{CC}$	$-\frac{2}{3}U_{CC}$	$\frac{1}{3}U_{CC}$	$\underline{U}_6 = \frac{2}{3}U_{CC} \cdot e^{j\frac{5}{3}\pi}$
ein	aus	ein	aus	ein	aus	0	0	0	$\underline{U}_7 = 0$
aus	ein	aus	ein	aus	ein	0	0	0	$\underline{U}_8 = 0$

Tabelle 3.1: zulässige Schaltkombinationen für einen Umrichter nach Abb. 3.1 und die korrespondierenden Spannungs-RMZ

Um dies zu bewerkstelligen, muss zuerst bestimmt werden, für welchen Winkel von $\underline{\Psi}_S(t)$ welcher Spannungs-RMZ $|\underline{\Psi}_S(t)|$ erhöht bzw. verringert und welcher Spannungs-RMZ das Drehmoment erhöht bzw. verringert. Um die Änderung von $|\underline{\Psi}_S(t)|$ zu berechnen, kann Gl. 2.12 herangezogen werden. Kann der Beitrag von $\underline{I}_S(t)$ vernachlässigt werden, da R klein ist, ergibt sich folgende Berechnungsvorschrift für die Änderung des Statorfluss-RMZ:

$$\frac{d\underline{\Psi}_S}{dt}(t) = \underline{U}_S(t) \tag{3.1}$$

Die Auswirkungen verschiedener Spannungs-RMZ auf das Drehmoment können mit dem Ergebnis aus Gl. 3.1 über Gl. 2.14 bestimmt werden. Die grafische Darstellung der verschiedenen RMZ ist in Abb. 3.2 abgebildet. Da die Auswirkungen der einzelnen Spannungs-RMZ auf $\underline{\Psi}_S(t)$ und M nicht für jeden Winkel von $\underline{\Psi}_S(t)$ die gleichen sind, wird die komplexe Ebene in die Sektoren $S_1 - S_6$ aufgeteilt. Innerhalb dieser Sektoren sind die Statorfluss auf-/abbauenden und die Drehmoment auf-/abbauenden Spannungs-RMZ stets dieselben. Aus diesen Überlegungen kann die Schalttabelle 3.2 abgeleitet werden. In Abb. 3.3 ist der Signalflussplan der DTC-Regelung angegeben. In der linken oberen Ecke ist Gl. 2.12 dargestellt. Durch Integration erhält man den Statorfluss-RMZ, aus dem dann sein Winkel und sein Betrag berechnet werden können. In der Mitte ist Gl. 2.14 dargestellt. Das berechnete Drehmoment wird mit dem Drehmoment-Sollwert verglichen und das Fehlersignal auf einen Dreipunkt-Hystereseregler gegeben.

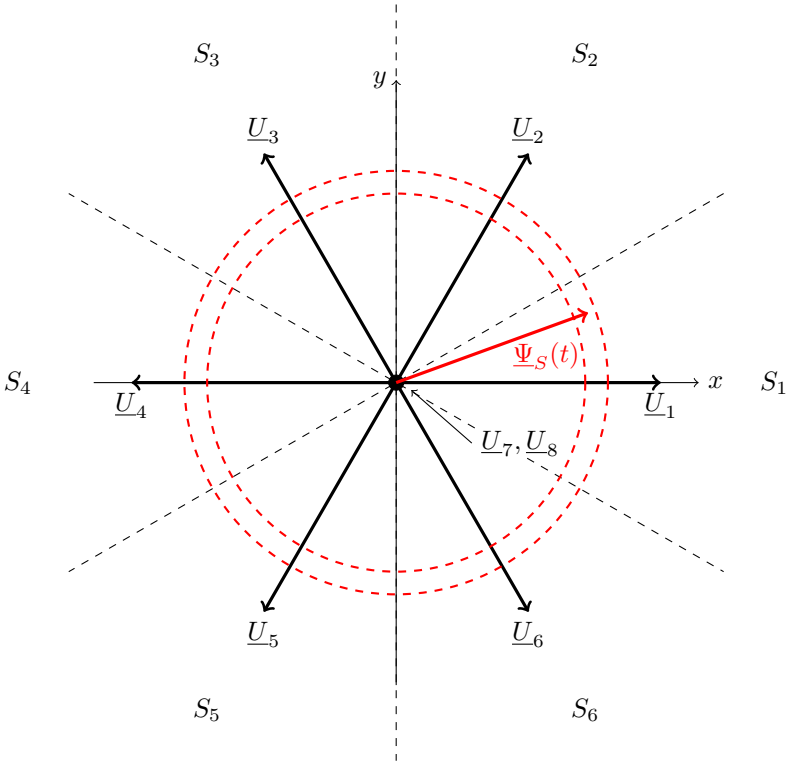


Abbildung 3.2: Spannungs-RMZ $\underline{U}_1 - \underline{U}_8$ des Umrichters und Sektoren $S_1 - S_6$ für das DTC-Verfahren. Die rot gestrichelten Kreise schließen den Bereich ein, in dem sich $\underline{\Psi}_S(t)$ bewegt.

$\underline{\Psi}_S$	M	S_1	S_2	S_3	S_4	S_5	S_6
aufbauen	aufbauen	$\underline{U}_2$	$\underline{U}_3$	$\underline{U}_4$	$\underline{U}_5$	$\underline{U}_6$	$\underline{U}_1$
aufbauen	abbauen	$\underline{U}_7$	$\underline{U}_8$	$\underline{U}_7$	$\underline{U}_8$	$\underline{U}_7$	$\underline{U}_8$
aufbauen	schnell abbauen	$\underline{U}_6$	$\underline{U}_1$	$\underline{U}_2$	$\underline{U}_3$	$\underline{U}_4$	$\underline{U}_5$
abbauen	aufbauen	$\underline{U}_3$	$\underline{U}_4$	$\underline{U}_5$	$\underline{U}_6$	$\underline{U}_1$	$\underline{U}_2$
abbauen	abbauen	$\underline{U}_8$	$\underline{U}_7$	$\underline{U}_8$	$\underline{U}_7$	$\underline{U}_8$	$\underline{U}_7$
abbauen	schnell abbauen	$\underline{U}_5$	$\underline{U}_6$	$\underline{U}_1$	$\underline{U}_2$	$\underline{U}_3$	$\underline{U}_4$

Tabelle 3.2: Schalttabelle für das DTC-Regelverfahren.

Das Gleiche wird auch für den Betrag des Statorflusses gemacht. Das Fehlersignal wird hier jedoch auf einen Zweipunkt-Hystereseregler gegeben. Es können auch andere Hystereseregler verwendet werden, jedoch kann sich dadurch der Aufbau von Tabelle 3.2 verändern. Hier wurde die Reglerauswahl aus [15] übernommen. Die Ausgangssignale der Hystereseregler werden zusammen mit dem Ergebnis der Sektorbestimmung für die Auswahl des richtigen Spannungs-RMZ aus Tabelle 3.2 verwendet. $\underline{I}_S$ wird aus den gemessenen Phasenströmen $I_a(t)$, $I_b(t)$, und $I_c(t)$ (Abb. 3.1) über Gl. 2.4 berechnet.

Über Gl. 2.15 kann durch Integration aus dem berechneten Drehmoment die Kreisfrequenz $\omega_R(t)$ des Rotors geschätzt werden. Wird $\omega_R(t)$ mit einem Sollwert $\omega_{R,soll}$ verglichen, so kann das Fehlersignal für die Berechnung von M_{soll} verwendet werden. Nach [4] kann für die Berechnung von M_{soll} ein PI-Regler verwendet werden. $|\underline{\Psi}_S|_{soll}$ kann über die Motorparameter wie folgt abgeschätzt werden [4]:

$$|\underline{\Psi}_S|_{soll} = \sqrt{\frac{4 \cdot L^2 \cdot L_R}{3 \cdot p \cdot M^2}} \tag{3.2}$$

- L_R: Induktivität des Rotors

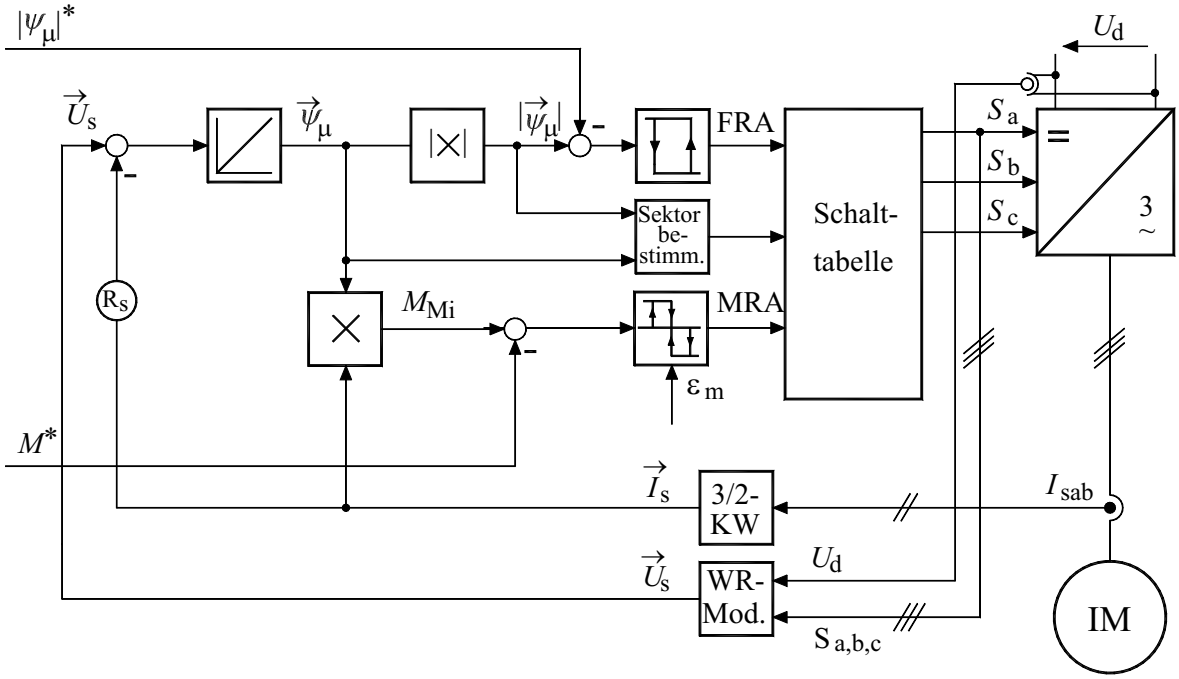


Abbildung 3.3: Signalflussplan der DTC-Regelung (Quelle: [15]). Notation: $\vec{U}_S = \underline{U}_S$, $\vec{I}_S = \underline{I}_S$, $\vec{\Psi}_\mu = \underline{\Psi}_S$, $|\vec{\Psi}_\mu|^* = |\underline{\Psi}_S|_{soll}$, $M_{Mi} = M$, $U_d = U_{CC}$, $M^* = M_{soll}$, $R_s = R$

3.2 Ableitung des Regelalgorithmus

Aus Abb. 3.3 kann nun der Algorithmus für den Prozessor abgeleitet werden. In Abb. 3.4 ist der resultierende Programmablaufplan (PAP) abgebildet. Am Anfang wird das Programm mit den Anfangssollwerten, den Motorparametern und dem Spannungs-RMZ zum Zeitpunkt $t = 0$ initialisiert. Anschließend wird der Sollwert für die Kreisfrequenz des Rotors eingelesen und anschließend die benötigten Größen nacheinander berechnet. Nachdem alle Eingangsignale für die Schalttabelle vorliegen, wird der entsprechende Spannungs-RMZ ausgewählt und dann die Endstufen angesteuert. Bei dieser sequentiellen Abarbeitung ist es wichtig einen Prozessor zu verwenden, der eine ausreichend hohe Taktrate besitzt. Je länger der Prozessor benötigt um alle Größen zu berechnen, desto geringer ist die maximale Rotationsfrequenz des Motors. Auf diese Problematik wird detaillierter in Kapitel 5 eingegangen.

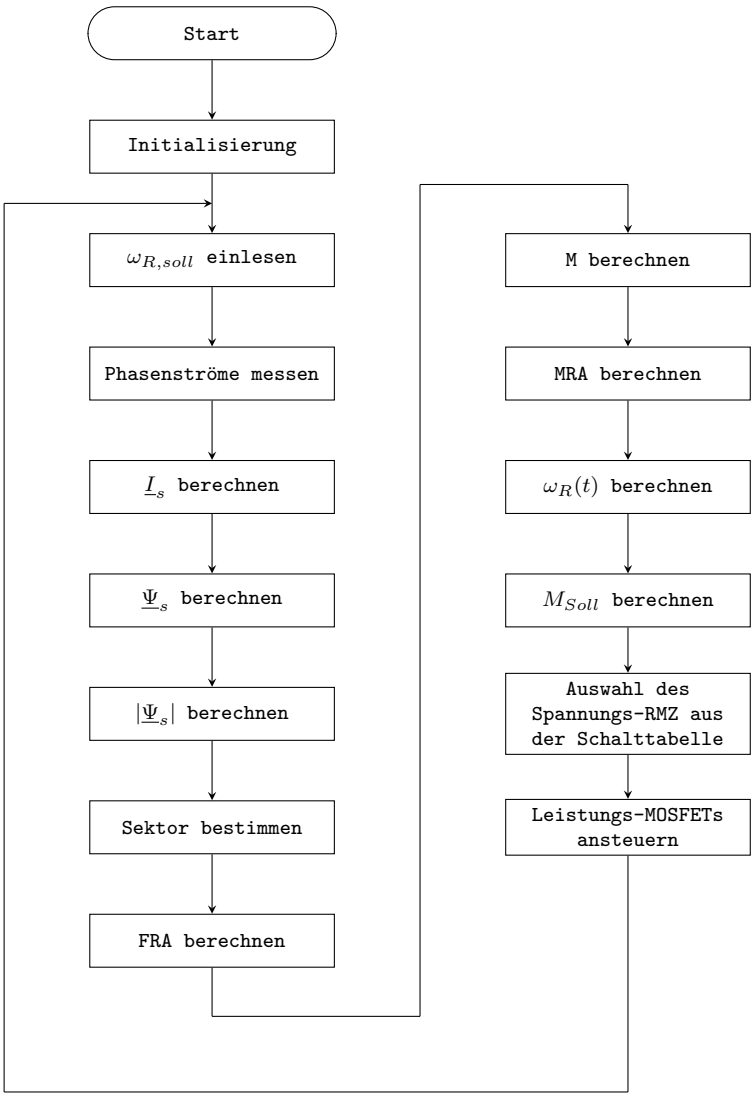


Abbildung 3.4: PAP des DTC-Regelverfahrens.

4 Realisierung der Hardware

In diesem Kapitel wird die Realisierung des Umrichters erläutert. Die Grundstruktur des Umrichters entspricht Abb. 3.1. Zuerst wird der Motor beschrieben, der für diese Thesis verwendet wird. Aus dem verwendeten Motor ergeben sich wichtige Eckdaten für den Schaltungsaufbau und für das Layout der Paltine. Anschließend folgt die Beschreibung der Endstufen. Hier wird vor allem auf die Gate-Beschaltung der Leistungs-MOSFETs eingegangen. Sie ist zentral für die korrekte Funktionsweise der Endstufen. Dem folgt eine Beschreibung des Mikroprozessors, der hier Verwendung findet. Abschließend werden die Strommessschaltungen beschrieben, über die die Phasenströme gemessen werden.

4.1 Der Motor

Für diese Arbeit wird der Wechselstrommotor DR62.0x60-2 der Firma Dunkermotoren GmbH verwendet. Er kann als Sternschaltung und als Dreieckschaltung betrieben werden. Außerdem besitzt er einen zugänglichen Sternpunkt, was Messungen am Motor vereinfacht. Der Motor wird in dieser Arbeit als Sternschaltung betrieben. Die Motor-Daten sind in Tabelle 4.1 aufgeführt. Eine 3D-Ansicht des Motors ist in Abb. 4.1 zu sehen. Oben auf dem Motor befindet sich die Abdeckung für den Sternpunkt und für die Anschlüsse der Motorphasen. Über die linke Öffnung der Abdeckung werden die Kabel für die drei Motorphasen und die Erdung des Motorgehäuses eingeführt. Rechts befindet sich die Motorwelle.

Bezeichnung	Formelzeichen	Wert
Nennspannung	U_N	$400V$
Nennfrequenz	f_N	$50Hz$
Nennstrom Phase a	I_{aN}	$0,25A \pm 10\%$
Nennleistungsfaktor	$cos(\varphi_N)$	$0,73 - 10\%$
Nenndrehmoment	M_N	$24Ncm$
Nenndrehzahl	n_N	$2600min^{-1} \pm 10\%$
Nennleistung	P_N	$65W$
Nennwirkungsgrad	η_N	$50,7 - 10\%$
Leerlaufstrom Phase a	I_{aL}	$0,25A \pm 10\%$
Leerlaufdrehzahl	n_L	$2980min^{-1} \pm 3\%$
Anlaufstrom Phase a	I_{aA}	$0,72A \pm 10\%$
Anlaufdrehmoment	M_a	$48Ncm \pm 15\%$
elektrischer Widerstand einer Phase	R	$164\Omega \pm 10\%$
elektrischer Widerstand von Phase a zu Phase b (Sternschaltung)	$R_{\curlywedge}$	$327\Omega \pm 10\%$
Massenträgheitsmoment des Rotors	J	$290gcm^2 \pm 10\%$
Polpaarzahl	p	1
Reibungsdrehmoment	M_R	$0,4Ncm$
Kippdrehmoment	M_K	$38,5Ncm \pm 10\%$
Aufbau des Rotors	-	18xAlu D31,48

Tabelle 4.1: Auszug aus den Motordaten für den Betrieb des Motors als Sternschaltung. Vollständiges Datenblatt: [6]

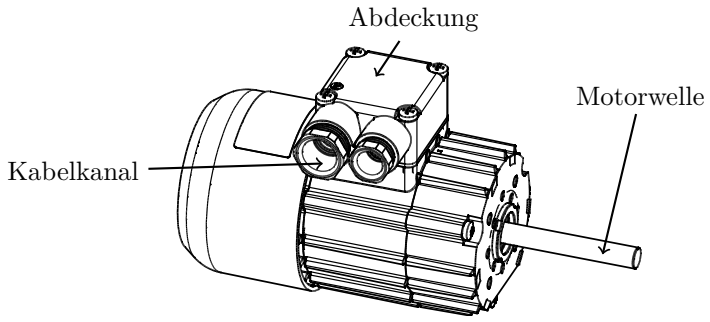


Abbildung 4.1: 3D-Ansicht des DR62.0x60-2 (Quelle der Originalgrafik: [2], Beschriftung hinzugefügt)

4.2 Aufbau der Endstufen

Wie in Abb. 3.1 bereits dargestellt, besteht eine Endstufe des Umrichters hier aus zwei übereinander angeordneten N-Kanal-MOSFETs. Die Endstufe soll die angeschlossene Motorphase entweder auf das Potential von U_{CC} anheben oder auf das Potential von GND absenken. Dabei muss eine gleichzeitige Aktivierung der beiden MOSFETs verhindert werden, da sonst ein Kurzschluss erzeugt wird. Außerdem besteht bei der Verwendung von zwei N-Kanal-MOSFETs das Problem, dass für den oberen MOSFET eine Spannung zwischen Gate und Source nicht ohne Weiteres erzeugt werden kann. Jedoch hat dieser Aufbau der Endstufe den Vorteil, dass, bezogen auf die MOSFETs, für beide Zustände der Endstufe gleiche Schalt- und Verzögerungszeiten vorliegen. Zusätzlich müssen auch die minimale und maximale Schaltfrequenz der Endstufe berücksichtigt werden. Um hohe Schaltfrequenzen der Endstufe zu ermöglichen müssen die folgenden beiden Bedingungen erfüllt werden:

1. die verwendeten MOSFETs müssen für diese Schaltfrequenzen ausgelegt sein
2. die Gate-Beschaltung muss schnell genug das Gate auf- bzw. entladen können

Ein Aufbau einer Endstufe, der die beschriebenen Anforderungen erfüllen soll, ist in Abb. 4.2 abgebildet. In den folgenden Abschnitten werden die Funktionsweise und die Dimensionierung der Bauteile erklärt.

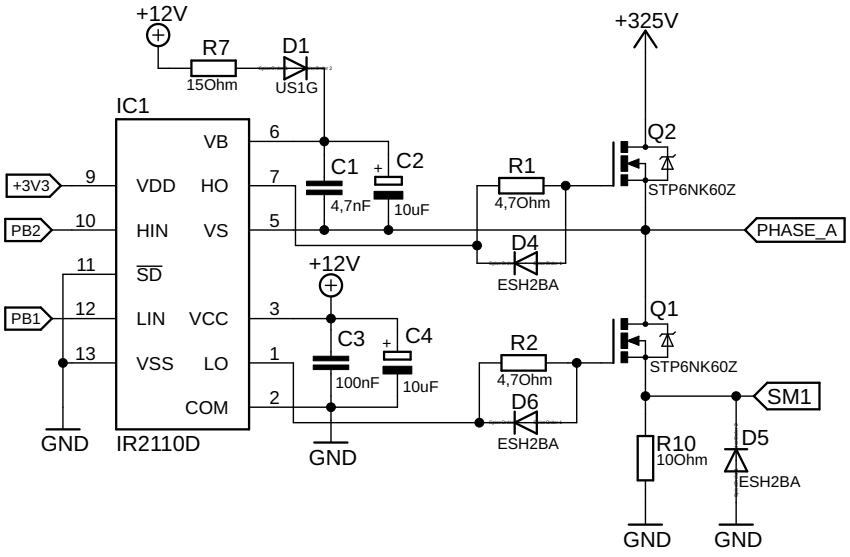


Abbildung 4.2: Aufbau einer Endstufe des Umrichters

4.2.1 N-Kanal-MOSFETs

Bei der Auswahl der MOSFETs Q_1 und Q_2 ist vor allem die hohe Versorgungsspannung $U_{CC} = 325V$ und die benötigte maximale Schaltfrequenz des Umrichters zu berücksichtigen. Des Weiteren sollte der Drain-Source-Widerstand im eingeschalteten Zustand $R_{DS,on}$ gering sein, um die Verlustleistung zu minimieren. Für die Endstufen wird der MOSFET STP6NK60Z von ST verwendet. Die wichtigsten Eckdaten sind in

Tabelle 4.2 aufgelistet. Er besitzt u.a. eine integrierte Zener-Diode, die im Fall von Q_2 Spannungsspitzen auf den Motorphasen schnell abbauen soll. Diese Spannungsspitzen werden bei negativer Änderung des Stromes durch die Motorinduktivitäten über folgenden Zusammenhang verursacht:

$$U(t) = L \cdot \frac{dI}{dt}(t) \tag{4.1}$$

Im Fall von Q_1 sorgt die Zener-Diode für den Ausgleich von Spannungsdifferenzen zwischen GND und der Motorphase. Diese Spannungsdifferenzen werden, bei negativer Änderung des Stromes durch Q_1, durch die Induktivität der Leiterbahn zwischen dem Drain-Anschluss von Q_1 und dem Knotenpunkt (zwischen Q_1 und Q_2) nach Gl. 4.1 verursacht. Dabei wird die Motorphase auf ein Potential unterhalb von GND gezogen. Die negative Änderung des Stromes kann dabei durch die folgenden beiden Schaltvorgänge verursacht werden:

- Q_1 wird abgeschaltet
- der obere MOSFET der Endstufe wird abgeschaltet, der eine Motorphase auf U_{CC} angehoben hat

Eine ausführlichere Beschreibung dieser Vorgänge kann [12] entnommen werden. Um diese Spannungsspitzen/Spannungsdifferenzen zu reduzieren, können die folgenden Maßnahmen ergriffen werden:

- Reduzierung der Induktivitäten (kann nur im Fall von Q_1 durchgeführt werden)
- Reduzierung von $\frac{dI}{dt}$ $\Rightarrow$ Erhöhung der Ein- und Ausschaltzeiten der MOSFETs (Näheres dazu in Abschnitt 4.2.2

Eine Reduzierung der parasitären Induktivitäten kann durch eine Minimierung der Abstände zwischen IC_1, Q_1 und Q_2 erreicht werden. In Abb. 4.3 ist das Layout einer Endstufe zu sehen. Das vollständige Layout der Platine ist in Unterkapitel 8.2 angegeben. Die MOSFETs wurden nache beieinander platziert. Der Abstand zwischen IC_1 und den MOSFETs wurde ebenfalls minimiert. Beim Erstellen des Layouts der Endstufen müssen aufgrund der hohen Versorgungsspannung Mindestabstände zwischen Bauteilen und zwischen Leiterbahnen berücksichtigt werden. Nach [8] beträgt der Mindestabstand der Leiterbahnen bei 320V 0,75mm, bei 400V 1mm. Für das Layout wurde für die Leiterbahnen und Bauteile, die die hohe Versorungsspannung führen können, eine Mindestkriechstrecke von 1,5mm angesetzt.

Bezeichnung	Formelzeichen	Wert
maximale Drain-Source-Spannung	$U_{DS,max}$	$600V$
maximale Gate-Source-Spannung	$U_{GS,max}$	$\pm 30V$
maximale Gate-Source-Schwellenspannung	$U_{GS,th,max}$	$4,5V$
maximaler kontinuierlicher Drain-Strom	$I_{D,max}$	$6A$
maximaler Drain-Source-Widerstand (MOSFET eingeschaltet)	$R_{DS,on,max}$	$1,2\Omega$
maximale Gesamtladung des Gates	Q_{ges}	$46nC$
maximale Schaltfrequenz	f_{max}	$1MHz$
typische Verzögerungszeit beim Einschalten	$t_{d,on}$	$14ns$
typische Verzögerungszeit beim Ausschalten	$t_{d,off}$	$47ns$
typische Anstiegszeit	t_r	$14ns$
typische Abfallzeit	t_f	$19ns$
maximaler Strom durch die Zener-Diode	I_{SD}	$6A$
maximale Spannung in Durchlassrichtung (Zener-Diode)	U_{SD}	$1,6V$
maximale Sperrverzögerungszeit der Zener-Diode	t_{rr}	$445ns$

Tabelle 4.2: wichtige Eckdaten des verwendeten MOSFETs STP6NK60Z (Datenblatt: [19])

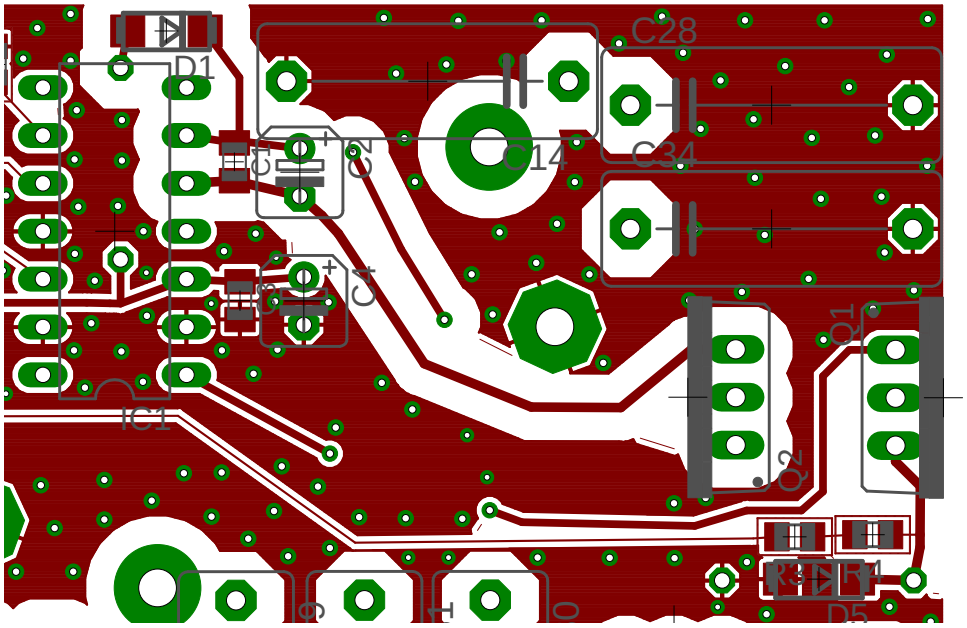


Abbildung 4.3: Layout einer Endstufe. Auf der linken Seite befindet sich IC_1, auf der rechten Seite Q_1 und Q_2.

4.2.2 Die Gate-Widerstände

Um die in Abschnitt 4.2.1 beschriebenen Spannungsspitzen und Spannungsdifferenzen zu minimieren, können die Schaltzeiten der MOSFETs erhöht werden. Dies kann über die Begrenzung des Gate-Stromes realisiert werden. Nach Gl. 4.2 erhöht ein geringerer Gate-Strom die Zeit, die benötigt wird um das Gate mit der erforderlichen Ladung aufzuladen. Eine solche Strombegrenzung kann durch eine Erhöhung der Gate-Widerstände R_1 und R_2 erreicht werden. Wird durch eine Erhöhung des Gate-Widerstandes z.B. die Ausschaltzeit des MOSFETs zu lang, kann durch eine Diode (D_4, D_6) parallel zum Gate-Widerstand eine Erhöhung der Ausschaltzeit verhindert werden. Beim Ausschalten des MOSFETs überbrücken diese beiden Dioden die Gate-Widerstände.

$$Q(t) = Q(t_0) + \int_{t_0}^{t} I(\tau)d\tau \tag{4.2}$$

4.2.3 Funktionsweise und Dimensionierung des Bootstrap-Kondensators

Wie am Anfang des Unterkapitels bereits erwähnt, kann nicht ohne Weiteres eine Gate-Source-Spannung für Q_2 erzeugt werden. Einer der gängigsten Lösungen für dieses Problem verwendet einen sog. Bootstrap-Kondensator (hier C_1 und C_2). Für den Fall, dass Q_1 eingeschaltet ist, wird der Pin V_s auf das Potential von GND gezogen. Damit fließt von der 12V-Versorgungsspannung aus ein Strom durch R_7 und D_1 und lädt die Kondensatoren C_1 und C_2 auf. Im nächsten Schritt wird Q_1 ausgeschaltet. Anschließend soll Q_2 eingeschaltet werden. Dazu zieht IC_1 den Pin H_O auf das Potential des Pins V_B. Ein Strom fließt von V_B durch die interne Endstufe von IC_1 über H_O und R_1 zum Gate von Q_2. Q_2 zieht die Motorphase (und damit auch den Pin V_S) allmählich auf das Potential von U_{CC}. Dabei sorgen C_1 und C_2 stets für eine positive Potentialdifferenz zwischen V_B und V_S. Sobald die Spannung über D_1 die Schwellenspannung, durch das ansteigende Potential von V_B, unterschreitet, sperrt D_1. Damit werden C_1 und C_2 von der 12V-Versorgungsspannung abgekoppelt. Sie können damit weiterhin die positive Potentialdifferenz zwischen V_B und V_S aufrecht erhalten und Q_2 bleibt eingeschaltet.
Für die korrekte Funktionsweise des Bootstrap-Kondensators sind ein paar wichtige Punkte zu beachten:

1. D_1 muss eine sehr kurze Sperrverzögerungszeit besitzen, da innerhalb dieser Zeitspanne Ladung von C_1 und C_2 über D_1 zurück in die 12V-Spannungsversorgung fließt.

2. D_1 muss **mindestens** die Spannung $U_{CC} + 12V$ blocken können!

3. Durch das Abschalten von Q_1 wird V_S auf ein Potential unterhalb von GND gezogen (siehe Abschnitt 4.2.1). Dadurch erhöht sich die Spannung, die an C_1 und C_2 anliegt. Es muss darauf geachtet werden, dass die Kondensatoren durch die erhöhte Spannung nicht überladen werden. Um das Überladen zu verhindern, kann die Kapazität erhöht werden [12]. Die Gesamtkapazität von C_1 und C_2 sollte mindestens $470nF$ betragen [12].

4. Um das Gate von Q_2 schnell aufladen zu können, sollte ein kleiner Keramik-Kondensator verwendet werden [12].

5. Um die Gate-Source-Spannung möglichst lange aufrechtzuerhalten, was vor allem bei niedrigen Schaltfrequenzen notwendig ist, muss die Kapazität des Bootstrap-Kondensators deutlich erhöht werden. Ein guter Kompromiss stellt eine Parallelschaltung aus Keramik-Kondensatoren und Elkos dar. Hier wird ein Keramik-Kondensator und ein Elko für die Realisierung des Bootstrap-Kondensators verwendet. Der Keramik-Kondensator hält die Ladung für das Gate vor und der Elko kompensiert die Leckströme.

Für D_1 wird hier die Diode US1G verwendet. Sie besitzt eine Sperrverzögerungszeit von maximal $50ns$ und kann dauerhaft eine Spannung von maximal $400V$ blocken [18]. Damit werden die oben aufgeführten Anforderungen an D_1 erfüllt. Um nun den Kapazitätswert des Bootstrap-Kondensators abschätzen zu können, kann nach [12] folgende Formel verwendet werden:

$$C_{BS} \geq \frac{2 \cdot \left(2 \cdot Q_{ges} + \frac{I_{QBS,max}}{f} + Q_{ls} + \frac{I_{CBS,leak}}{f}\right)}{U_{DD} - U_{D1} - U_{Q1} - U_{BS,min}} \tag{4.3}$$

Formelzeichen	Wert	Bezeichnung
Q_{ges}	$46nC$ (Tab. 4.2)	Gesamtladung des Gates
f	$20Hz$	Mindestschaltfrequenz des MOSFETs
$I_{CBS,leak}$	$0A$	Leckstrom des Bootstrap-Kondensators
$I_{QBS,max}$	$230\mu A$ [13]	maximaler Ruhestrom der internen Endstufe zwischen V_b und V_S
U_{DD}	$12V$	Spannungsversorgung
U_{D1}	$1V$ [18]	Spannung von D_1 in Durchlassrichtung
U_{Q1}	$2,5V$	Spannungsabfall über eingeschaltetem Q_1 und R_{10}. Angesetzt wurde ein Strom durch R_{10} von 200mA.
$U_{BS,min}$	$6V$	minimale Spannung zwischen V_b und V_S
Q_{ls}	$5nC$ [12]	Ladung die bei jedem Schaltvorgang intern in IC_1 für den Level-Shift benötigt wird.

Tabelle 4.3: Zahlenwerte und Erklärungen zu Formel 4.3

Mit den Zahlenwerten aus Tabelle 4.3 ergibt sich über Gl. 4.3 eine Kapazität des Bootstrap-Kondensators von $C_{BS} \geq 9\mu F$. Diese Kapazität wird in Form eines Elkos (C_2) realisiert. Um die Kapazität des Keramik-Kondensators abzuschätzen, kann folgende Formel herangezogen werden:

$$C = \frac{Q_{ges}}{U_{BS}} \tag{4.4}$$

Mit $Q_{ges} = 46nC$ und $U_{BS} = 11V$ ergibt sich für die Kapazität des Keramik-Kondensators ein Wert von $C \geq 4,18nF$. Damit werden auch die Anforderungen an den Bootstrap-Kondensator erfüllt. In Abb. 4.3 sind die Kondensatoren im Layout abgebildet. Sie wurden so nahe wie möglich am Gate-Treiber positioniert. C_3 und C_4 dienen der Glättung der 12V-Versorgungsspannung.

4.2.4 Der Gate-Treiber

Den Strom für das Laden der Gates stellt der Gate-Treiber IC_1 bereit. Da bei hohen Schaltfrequenzen der MOSFETs in kurzer Zeit ein hoher Gate-Strom benötigt wird, wird als Gate-Treiber der IC IR2110 von International Rectifier verwendet. In Tabelle 4.4 sind die wichtigsten Eckdaten des Gate-Treibers aufgelistet. Neben einem maximalen Ausgangsstrom von $2A$, besitzt er nahezu identische Verzögerungszeiten ($\pm 10ns$) für beide MOSFETs. Dies erleichtert die Ansteuerung erheblich. Es dürfen jedoch nie beiden Eingangssignale (H_{IN} und L_{IN}) gleichzeitig einen High-Pegel führen, da ansonsten beide MOSFETs gleichzeitig eingeschaltet werden und dadurch ein Kurzschluss erzeugt wird. Dieses Problem wird später über die Software gelöst. Die Eingangssignale H_{IN} und L_{IN} sind mit 3,3V-CMOS-Ausgängen kompatibel, womit der IC direkt von einem Mikroprozessor angesteuert werden kann. Das Blockschaltbild des Gate-Treibers ist in Abb. 4.4 zu sehen. Im oberen Bereich befindet sich die Logik und die Endstufe für die Ansteuerung von Q_2. Die untere Endstufe dient der Ansteuerung von Q_1. Es ist zu erkennen, dass die Beiden Endstufen unabhängig voneinander sind. Daher kann an der oberen Endstufe ein Bootstrap-Kondensator betrieben werden.

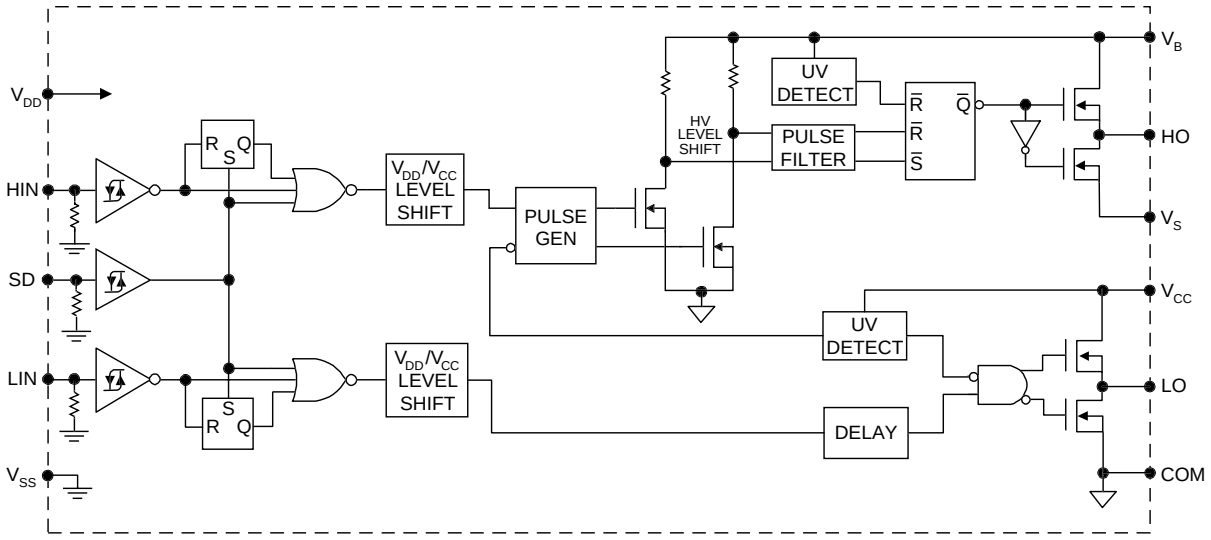


Abbildung 4.4: Blockschaltbild des Gate-Treibers (Quelle: [13])

Bezeichnung	Formelzeichen	Wert
maximale Versorgungsspannung für die obere Endstufe (bezogen auf COM)	$U_{B,max}$	$525V$
maximale Versorgungsspannung für die untere Endstufe (bezogen auf COM)	$U_{DD,max}$	$25V$
maximale Verzögerungszeit beim Einschalten	$t_{d,on}$	$150ns$
maximale Verzögerungszeit beim Ausschalten	$t_{d,off}$	$125ns$
maximale Anstiegszeit	t_r	$35ns$
maximale Abfallzeit	t_f	$25ns$
maximale zeitliche Abweichung zwischen den Verzögerungszeiten der oberen und der unteren Endstufe	Δt_d	$10ns$

Tabelle 4.4: Eckdaten des Gate-Treibers (Datenblatt:[13])

4.2.5 Shunt für die Strommessung

Für das in Kapitel 3 beschriebene Regelverfahren müssen die Phasenströme des Motors gemessen werden. Dazu wurde unterhalb von Q_1 ein Shunt (R_{10}) angebracht. Der Spannungsabfall über R_{10} dient als Eingangssignal für die Strommessschaltung. Die Verwendung eines Shunts bietet den Vorteil, dass auch Gleichströme einfach gemessen werden können, was vor allem bei niedrigen Drehzahlen des Motors von Vorteil ist. Um den Entwurf der Strommessschaltung zu vereinfachen, wurde R_{10} bewusst nicht in der Motorphase platziert. Damit werden eventuelle Probleme mit Spannungsspitzen und der allgemein hohen Spannung, die die Motorphase führen kann, umgangen. Um den Phasenstrom über R_{10} messen zu können, muss jedoch Q_1 eingeschaltet sein. Da aber nach Tabelle 3.1, abgesehen von $\underline{U}_7$, bei jedem Spannungs-RMZ es mindestens eine Endstufe gibt, bei der der untere MOSFET eingeschaltet ist, können die drei Phasenströme wie folgt berechnet werden:

1. **Voraussetzungen:** der Motor wird als Sternschaltung betrieben
2. Für die Spannungs-RMZ $\underline{U}_1$, $\underline{U}_3$, $\underline{U}_5$ und $\underline{U}_8$ können die Phasenströme über die Kontenregel berechnet werden, da für diese Spannungs-RMZ bei zwei Endstufen die unteren MOSFETs eingeschaltet sind.
3. Für die übrigen Spannungs-RMZ kann nur an einer Endstufe ein Phasenstrom gemessen werden. Damit kann die Knotengleichung nicht gelöst werden. Es soll jedoch der Versuch unternommen werden, einen Phasenstrom zu schätzen und den dritten Strom dann zu berechnen.

Die Diode D_5 wird benötigt, um den negativen Spannungsabfall über R_{10} zu begrenzen. Diese negative Spannung entsteht, wenn das Potential von V_S das Potential von GND unterschreitet (siehe Abschnitt 4.2.1). Dann fließt ein Strom von GND über die Zener-Diode von Q_1 nach V_S. Da dieser Strom sehr hoch sein kann (siehe dazu [12, Kapitel 6]), wurde vorsorglich eine Diode parallel zu R_{10} gesetzt, die die negative Spannung auf $-0,9V$ begrenzt.
R_{10} wurde als Metallschicht-Widerstand der Bauform 0207 realisiert. Die Widerstandstoleranz beträgt $\pm 0,1\%$. Da die Phasenströme im Betrieb $240mA$ nicht überschreiten, kann die maximale Verlustleistung von R_{10} über die Formel

$$P_{R10} = R_{10} \cdot I^2 \tag{4.5}$$

zu $0,576W$ berechnet werden. Damit liegt sie unterhalb der zulässigen maximalen Verlustleistung von $0,6W$.

4.3 Der Mikroprozessor

Als Mikroprozessor wird ein STM32F411RE verwendet. Er besitzt eine maximale Taktrate von $100MHz$, was für den Algorithmus nach Abb. 3.4 ausreichen sollte. Über einen integrierten 12-Bit-ADC können die Phasenströme gemessen werden. Der Prozessor befindet sich auf dem Evaluation-Board NUCLEO-F411RE (Näheres dazu in [21]). Damit kann der Prozessor sofort in Betrieb genommen werden. Außerdem kann das Evaluation-Board über die Buchsenleisten CN7 und CN10 (Abb. 8.1e und Abb. 8.1f) einfach in die Schaltung integriert werden. Ein weiterer Grund für die Verwendung eines STM32-Prozessors besteht darin, dass bereits in früheren Projekten mit STM32-Prozessoren gearbeitet wurde. Somit entfallen die Einarbeitungszeiten in die Entwicklungsumgebung und in die Software-Bibliotheken.

4.4 Aufbau der Strommessschaltungen

Die Strommessschaltung hat die Aufgabe, den Spannungsabfall über R_{10} soweit aufzubereiten, dass er über einen ADC-Eingang des Mikroprozessors eingelesen werden kann. Bei der Realisierung der Messschaltung müssen folgende Randbedingungen berücksichtigt werden:

- Die Spannung am ADC-Eingang des Prozessors muss im Bereich $0 \leq U_{ADC} \leq 3,3V$ liegen [20].
- Die Messschaltung soll über den gesamten Spannungsbereich möglichst linear sein.
- Die Messschaltung soll eine Genauigkeit von mindestens $\pm 1mA$ aufweisen.
- Die Messschaltung muss Frequenzen bis mind. $10kHz$ ohne übermäßige Amplitudenverzerrung und Phasenverschiebung übertragen können.

Um diese Anforderungen zu erfüllen, kann die Messschaltung gemäß Abb. 4.5 aufgebaut werden. Dabei wird zur Erzeugung einer hohen Linearität ein Optokoppler mit Rückkopplungszweig (OK_1) verwendet. Liegt an SM_1 eine positive Spannung an, wird er Ausgang von OPV_1 auf ein Potential unterhalb von GND gezogen und die interne LED strahlt eine gewisse Menge Licht ab. Damit beginnt in den beiden Fotodioden ein Strom zu fließen (I_{PL} für die linke , I_{PR} für die rechte Fotodiode). Da I_{PL} durch R_3 und R_4 begrenzt wird, baut sich über der linken Fotodiode eine Spannung auf. OPV_1 verstärkt diese Spannung entsprechend und passt damit den Strom durch die LED an. Über diese Rückkopplung werden die Nichtlinearitäten und Temperaturabhängigkeiten der LED kompensiert [22] und die Höhe der Eingangsspannung auf die Stärke des Stromes I_{PR} abgebildet. OPV_2 dient zur Verstärkung der durch I_{PR} erzeugten Spannung. Der Ausgang von OPV_2 wird mit dem ADC-Eingang verbunden. Hohe Eingangsspannungen werden durch die Kennlinie der linken Fotodiode begrenzt und nicht an den Ausgang weitergegeben. Aufgrund des Aufbaus des Optokopplers werden negative Spannungen blockiert. Für die Berechnung der Widerstandswerte werden nach [22] folgende Formeln verwendet:

$$R_x = R_3 + R_4$$
$$R_y = R_{12} + R_{49} + R_{50}$$
$$I_{PL,max} = K_1 \cdot I_{LED,max} \tag{4.6}$$
$$R_x = \frac{U_{SM1,max}}{I_{PL,max}} \tag{4.7}$$
$$R_y = \frac{R_x \cdot G}{K_3} \tag{4.8}$$

Formelzeichen	Wert	Bezeichnung
$I_{LED,max}$	$40mA$	maximaler Strom durch die LED
K_1	$0,0075$ [23, Fig.3]	Verstärkungsfaktor zwischen I_{PL} und I_{LED}
$U_{SM1,max}$	$2,5V$	maximale Eingangsspannung ($R_{10} \cdot 250mA$)
G	$1,3$	Gesamtverstärkung zwischen Eingangsspannung und Ausgangsspannung
K_3	$0,93$ [23]	Verstärkung des Optokopplers

Tabelle 4.5: Zahlenwerte und Erklärungen zu Gl. 4.6 - 4.8

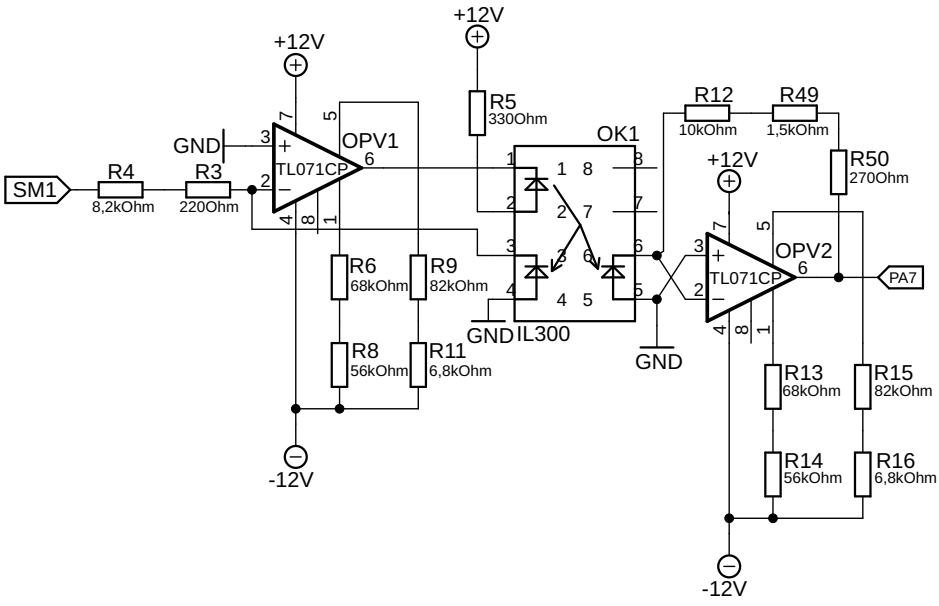


Abbildung 4.5: Aufbau der Strommessschaltungen (Schaltungsaufbau aus [23] entnommen)

Mit den Zahlenwerten aus Tabelle 4.5 ergeben sich folgende Widerstandswerte: $R_x = 8,4k\Omega$ und $R_y = 11,74k\Omega$. Die Widerstände R_6, R_8, R_9, R_{11} und R_{13} bis R_{16} werden für die Korrektur des Offsets von OPV_1 und OPV_2 benötigt.

4.5 Spannungsversorgung

Wie bereits in Kapitel 1 erwähnt, soll der ASM mit $325V$ Gleichspannung betrieben werden. Aufgrund eines defekten Spannungsmoduls konnte die Spannungsversorgung nicht wie geplant realisiert werden. Die alternative Spannungsversorgung sieht die Verwendung von zwei Labornetzteilen vor. Folgende Labornetzteile werden verwendet:

- GwINSTEK GPS-4303: Wird für die Spannungsversorgung der Endstufen (U_{CC}) verwendet. Dabei werden zwei Kanäle seriell betrieben, wodurch eine maximale Spannung von $U_{CC} = 60V$ zur Verfügung steht.
- Agilent E3631A: Wird für die 12V- und die -12V-Spannungsversorgung verwendet.

Um eine Stromfluss in die Spannungsversorgung hinein zu verhindern, wurden Sperrdioden verbaut. In Abb. 4.6 sind verwendeten Dioden abgebildet. Für die Sperrdiode in der 325V-Versorgungsleitung (D_{13}) musste eine größere Diode verwendet werden, da der Gesamtstrom den zulässigen Maximalstrom der Diode US1G überschreitet.

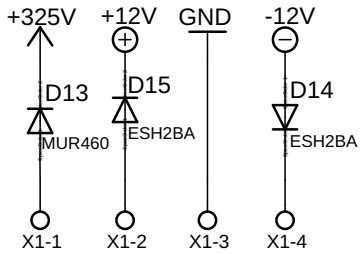


Abbildung 4.6: Sperrdioden für die Spannungsversorgungen

5 Realisierung der Software

Durch die Software wird der Regelalgorithmus auf dem Prozessor implementiert. Die Software sollte daher möglichst viele Prozesse parallelisiert, um die Gesamtberechnungszeit möglichst kurz zu halten. Dazu müssen u.a. der ADC und die Direct Memory Access (DMA)-Funktion richtig konfiguriert werden. Dazu wird die Software STM32CubeMX von ST verwendet, die über eine grafische Oberfläche die Konfiguration des Prozessors vereinfacht. Als Entwicklungsumgebung wird uVision5 von Keil verwendet.

5.1 Implementierung des geplanten Regelalgorithmus

Der Regelalgorithmus nach Abb. 3.4 benötigt den Drehzahl-Sollwert und die Messwerte aus den Strommessschaltungen um die restlichen Größen berechnen zu können. Dazu wurden die ADC-Kanäle 7 (= $PA7$), 4 (= $PA4$), 8 (= $PB0$) und 6 (= $PA6$) mit einer Auflösung von 12 Bit und einer Abtastzeit von 480 Zyklen konfiguriert (Unterkapitel 8.4, Zeile 267-341). Um die Auslastung des Prozessors zu verringern, werden die Messwerte in den ADC-Registern über die DMA-Funktion in das vordefinierte Array

Quellcode 5.1: Array für die ADC-Messerte

```
uint32_t adcData[4];                                                    1
```

geschrieben. Nachdem dieser Vorgang beendet wurde, wird ein Interrupt ausgelöst und die Callback-Funktion

Quellcode 5.2: ADC-Callback-Funktion

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)                  1
{                                                                       2
        .....                                                           3
}                                                                       4
```

aufgerufen. Hier können die Messwerte konvertiert und angepasst werden. Anschließend können die anderen Größen berechnet werden.
Bereits bei Zwischentests der Software hat sich jedoch gezeigt, dass die Berechnungen für den Regelalgorithmus nach Abb. 3.4 viel zulange dauern (ungefähr 300ms). Aufgrund des engen Zeitplans wurde das DTC-Regelverfahren nicht weiter verfolgt. Die möglichen Ursachen und Optimierungsmöglichkeiten werden in Kapitel 7 diskutiert. Stattdessen wird ein deutlich einfacheres Verfahren implementiert, das aus dem DTC-Regelverfahren abgeleitet werden kann.

5.2 Implementierung des alternativen Steuerverfahrens

Die Grundidee dieses Steuerverfahrens besteht darin, eine fest vorgegebene Abfolge von Spannungs-RMZ nacheinander zu schalten. Dieses Verfahren wird auch als Grundfrequenztaktung bezeichnet [15]. Dabei wird die Motordrehzahl über die Länge der Verzögerungszeit zwischen dem Schalten der einzelnen Spannungs-RMZ gesteuert. Die Abfolge der Spannungs-RMZ entspricht der ersten Zeile in Tabelle 3.2. Werden in dieser Reihenfolge die Spannungs-RMZ geschaltet, werden rechteckige Phasenspannungen erzeugt, die untereinander um 120° versetzt sind. Für eine Rotation des Motors im Uhrzeigersinn wird die Zeile von links nach rechts zyklisch abgearbeitet, für eine Rotation der Motors gegen den Uhrzeigersinn von rechts nach links. Bei konstantem Lastdrehmoment stellt sich die Drehzahl des Motors ein, bei der ein Drehmoment-Gleichgewicht herrscht. Die Strommessschaltungen werden hier nicht mehr benötigt.
Die Grundstruktur der Software ist in Abb. 5.1 angegeben. Asynchron zur Endlos-Schleife wird der Spannungswert des Potis in die Verzögerungszeit umgerechnet und die gewünschte Drehrichtung bestimmt. Bei der Implementierung der einzelnen Spannungs-RMZ muss auf die Reihenfolge geachtet werden, mit der die MOSFETs angesteuert werden. Im Quellcode-Ausschnitt 5.3 ist die Implementierung für $\underline{U}_1$ angegeben. Es werden zuerst die MOSFETs abgeschaltet (Zeile 4-6), die für den betreffenden Spannungs-RMZ inaktiv sein sollen und anschließend werden die anderen drei MOSFETs aktiviert (Zeile 9-11). Dieses Vorgehen verhindert die Erzeugung eines Kurzschlusses in den Endstufen. Die einzelnen RMZ-Funktionen werden über Funktionszeiger in einem Array in der entsprechenden Reihenfolge zusammengefasst. Die

Implementierung dieses Arrays ist im Quellcode-Ausschnitt 5.4 angegeben. In Zeile 3 wird die Reihenfolge für eine Rotation gegen den Uhrzeigersinn definiert, in Zeile 4 für eine Rotation im Uhrzeigersinn. Die Konfiguration des ADCs und der DMA-Funktion wird aus Unterkapitel 5.1 übernommen. Über das Poti R_{43} (Abb. 8.1g) wird die gewünschte Drehzahl eingestellt.

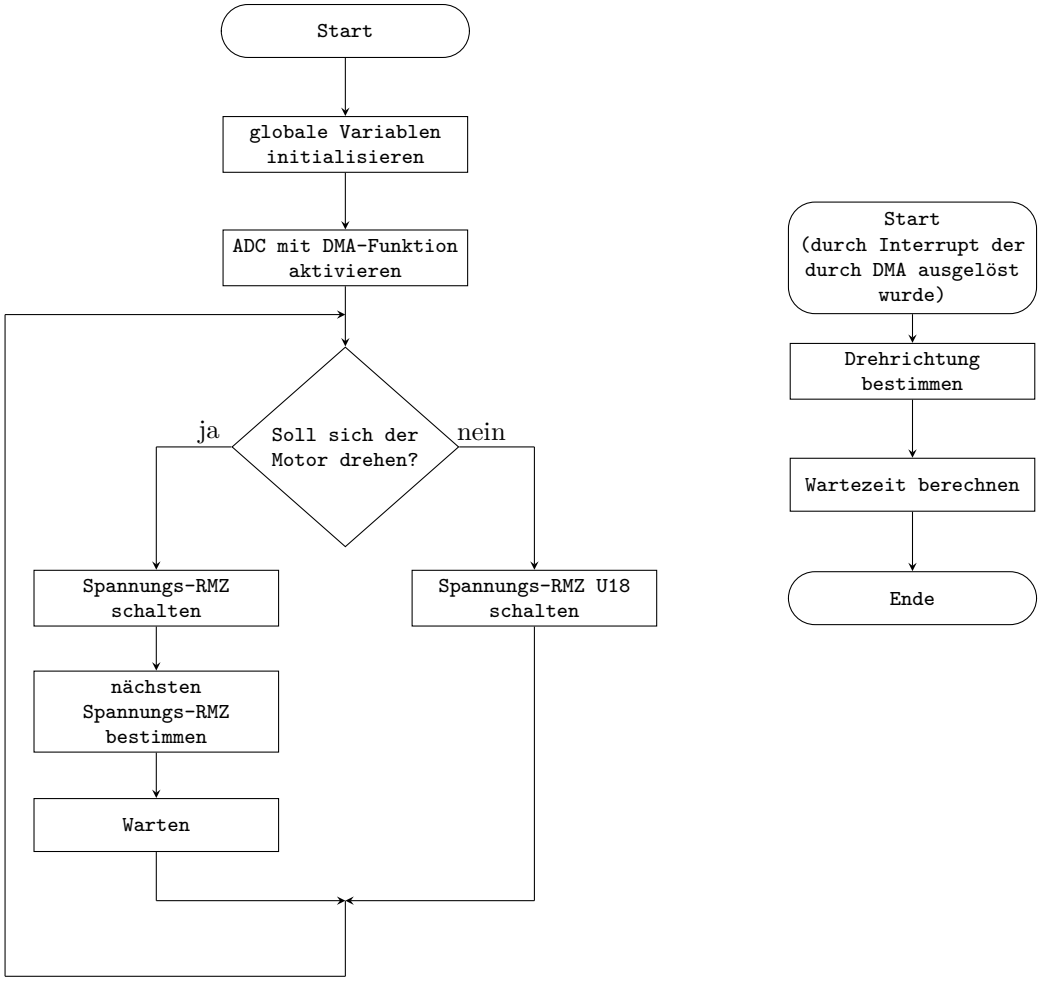


Abbildung 5.1: PAP des Steueralgorithmus. Links die Endlos-Schleife, rechts die Berechnung der Verzögerungszeit.

Quellcode 5.3: Implementierung eines Spannungs-RMZs

```
void U11(void)                                                                    1
{                                                                                 2
        //MOSFETs abschalten                                                      3
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_RESET); //MOSFET Q1           4
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_3,GPIO_PIN_RESET); //MOSFET Q4           5
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_3,GPIO_PIN_RESET); //MOSFET Q6           6
                                                                                  7
        //MOSFETs einschalten                                                     8
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_2,GPIO_PIN_SET); //MOSFET Q2            9
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_SET); //MOSFET Q3            10
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_2,GPIO_PIN_SET); //MOSFET Q5            11
}                                                                                 12
```

Quellcode 5.4: Implementierung der Abfolge der RMZ

```
typedef void (*spRMZ) (void);                                                     1
                                                                                  2
spRMZ swTable[2][6]={{U11,U16,U15,U14,U13,U12},                                   3
                     {U11,U12,U13,U14,U15,U16}};                                  4
```

In der ADC-Callback-Funktion wird nun der Messwert aus ADC-Kanal 6 in die Verzögerungszeit umgerechnet. Die Berechnung ist im Quellcode-Ausschnitt 5.5 zu sehen. Um die Rotation in beide Richtungen und den Stillstand des Motors umzusetzen, werden globale Variablen benötigt. Diese werden in den Zeilen 1-12 angelegt. Anschließend wird in Zeile 21 die Verzögerungszeit für die Rotation gegen den Uhrzeigersinn und in Zeile 28 die Verzögerungszeit für die Rotation im Uhrzeigersinn berechnet. Sollte sich das Poti in der Mittelstellung befinden, wird nur der Variable cw ein Wert zugewiesen. In Zeile 36 wurde noch eine Hysterese implementiert. Sie reduziert die Empfindlichkeit der Drehzahleingabe (Poti).

Quellcode 5.5: Berechnung der Verzögerungszeit

```
int k; //enspricht Verzögerungszeit zwischen den                    1
einzelnen Spannungs-RMZ                                             2
                                                                    3
int cw; //cw=1-> Rotation im Uhrzeigersinn,                         4
cw=0-> Rotation gegen den Uhrzeigersinn. cw=2-> stop                5
int cwBegin; //ADC-Wert ab dem der CW-Bereich beginnt               6
int ccwBegin;//ADC-Wert ab dem der CCW-Bereich beginnt              7
int cwOffset; //Offset für die Gerade im CW-Bereich                 8
int ccwOffset;//Offset für die Gerade im CCW-Bereich               9
int mcw; //Steigung der Gerade für den CW-Bereich                   10
int mccw; //Steigung der Gerade für den CW-Bereich                  11
int hysK; //Hysteresebreite für k                                   12
                                                                    13
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)              14
{                                                                   15
        int x=0;                                                    16
                                                                    17
        if(adcData[3]>ccwBegin)                                     18
        {                                                           19
                cw=0;                                               20
                x =mccw*(adcData[3]-ccwBegin)+ccwOffset;            21
        }                                                           22
        else                                                        23
        {                                                           24
                if(adcData[3]<cwBegin)                              25
                {                                                   26
                        cw=1;                                       27
                        x =mcw*adcData[3]+cwOffset;                 28
                }                                                   29
                else                                                30
                {                                                   31
                        cw=2;                                       32
                }                                                   33
        }                                                           34
                                                                    35
        if((k-x<-hysK)||(k-x>hysK)) k=x;                            36
}                                                                   37
```

In der main-Funktion (Quellcode-Ausschnitt 5.6) werden nun zuerst die globalen Variablen initialisiert (Zeile 10-16). Die Zahlenwerte wurden empirisch ermittelt und sind für jeden Motor verschieden. Sie passen die Spannungsbereiche des Potis an den Drehzahlbereich des Motors an. Anschließend wird der ADC gestartet (Zeile 18). In der Endlosschleife wird nun geprüft, ob sich der Motor überhaupt drehen soll (Zeile 24). Falls ja, wird, abhängig von cw, der i-te Spannungs-RMZ geschaltet (Zeile 26). Anschließend wird i für den nächsten Schleifendurchlauf um eins erhöht bzw. wieder auf Null gesetzt (Zeile 27-28). Zum Schluss wird der Programmablauf in Form einer for-Schleife verzögert. Hier kommt die im Quellcode-Ausschnitt 5.5 berechnete Verzögerungszeit k zum Einsatz. Für den Fall, dass sich der Motor nicht drehen soll, wird der Spannungs-RMZ $\underline{U}_8$ geschaltet (Zeile 33).
Der vollständige Quellcode ist in Unterkapitel 8.4 angegeben. Das vollständige Software-Projekt befindet sich auf dem beiliegenden Datenträger.

Quellcode 5.6: Implementierung der main-Funktion

```
int main(void)
{
  HAL_Init();
  SystemClock_Config();

  MX_GPIO_Init();
  MX_DMA_Init();
  MX_ADC1_Init();

  ccwBegin = 2150;
  cwBegin = 1950;
  mccw = -485;
  mcw = 485;
  ccwOffset = 1000000;
  cwOffset = 55000;
  hysK =50;

  HAL_ADC_Start_DMA(&hadc1,adcData,4); //ADC in DMA-Mode starten

  int i=0;

  while (1)
  {
        if(cw<2)
        {
              swTable[cw][i]();
              if(i<5) i++;
              else i=0;
              for(int j=0; j<k;i++);
        }
        else
        {
              U18();
        }
  }
}
```

6 Messergebnisse

In diesem Kapitel werden die Messergebnisse für die erstellte Hardware vorgestellt. Zu Beginn werden der Messaufbau und die Ergebnisse für die Endstufen diskutiert. Dabei werden Messungen bei verschiedenen Drehzahlen des Motors präsentiert und geprüft, ob die festgelegten Grenzen für die einzelnen Signale eingehalten werden. Anschließend werden die Ergebnisse für die Strommessschaltungen vorgestellt. Dabei werden die Linearität zwischen Ein- und Ausgangssignal und das Frequenzverhalten beurteilt.

6.1 Messergebnisse für die Endstufen

Für die folgenden Messungen wurde der Motor über die Endstufen, durch den in Unterkapitel 5.2 beschriebenen Algorithmus, angesteuert. Für die Messung der Signale wurde verwendet:

- Trenntransformator für das Oszilloskop: BLOCK Typ BR 802
- digitales Oszilloskop: RIGOL DS1104Z

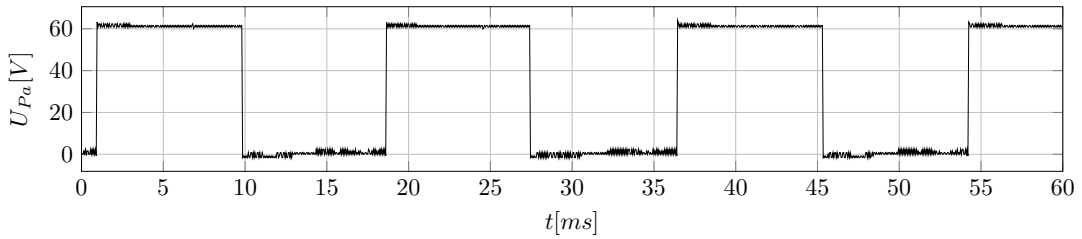
Beim Messen der Signale wurde darauf geachtet, dass die Masseleitung der Messspitzen so kurz wie möglich ist, um Störungen auf den gemessenen Signalen zu minimieren. Zusätzlich wurde das Oszilloskop über ein Trenntransformator betrieben, was zusätzlich Störsignale aus dem Netz minimiert.
In Abbildung 6.1 sind die gemessenen Signale für eine Drehzahl von $2700 min^{-1}$ abgebildet. Die Phasenspannungen U_{Pa} (Abb. 6.1a) und U_{Pc} (Abb. 6.1b) weisen keine Überschwinger auf. Auch U_{PB2} bewegt sich sauber zwischen 0 und $3,3V$. Dies ist wichtig, da ansonsten Beschädigungen des Prozessors hervorgerufen werden können. Für die Endstufe, über die Phase c geschaltet wird, ist in Abb. 6.1d das Ausgangssignal der Strommessschaltung aufgetragen. Sobald U_{Pc} abfällt, zeichnet sich in U_{PB0} der Verlauf des Phasenstromes ab. Das Signal bleibt ebenfalls innerhalb des angesetzten Wertebereichs. U_{CC} und U_{STM} sind ebenfalls konstant und bleiben auch innerhalb des zulässigen Bereichs, was besonders für U_{STM} wichtig ist, da diese Versorgungsspannung vom STM-Board abgegriffen wird.
In Abb. 6.2 sind die Messergebnisse für eine Drehzahl von $210 min^{-1}$ zu sehen. Die Spannungen der Motorphasen (Abb. 6.2a und 6.2b) weisen ebenfalls keine Überschwinger auf. Auch das Steuersignal U_{PB2} (Abb. 6.2c) bleibt in den definierten Grenzen. Der Phasenstrom (wird durch U_{PB0} (Abb. 6.2d) repräsentiert) hat sich erhöht, was sich auf eine deutlich längere Verzögerungszeit (im Vergleich zu Abb. 6.1) zurückführen lässt. Das Signal bleibt aber auch innerhalb der festgesetzten Grenzen. Auf den Versorgungsleitungen sind keine Störungen zu erkennen (Abb. 6.2e, 6.2f).

6.2 Messergebnisse für die Strommessschaltungen

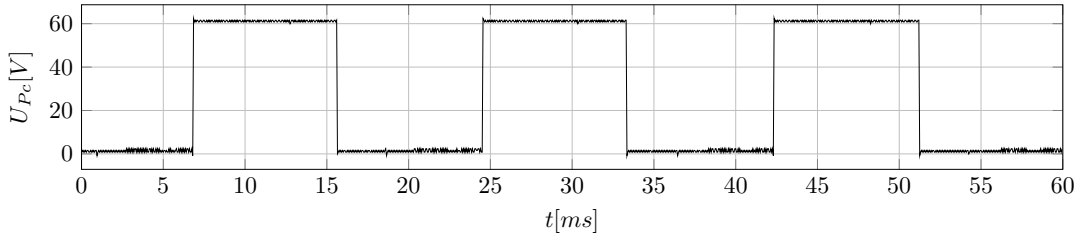
Um den Zusammenhang zwischen Ein- und Ausgangssignal zu messen, wurde über ein Netzteil eine Gleichspannung am Eingang der Messschaltung angelegt und anschließend die Ausgangsspannung gemessen. Für die Messung des Frequenzverhaltens wurde am Eingang ein Frequenzgenerator angeschlossen und das Verhalten des Ausgangs über ein Oszilloskop gemessen. Dabei wurde das Oszilloskop ebenfalls über einen Trenntransformator betrieben.
Für die Messungen wurde verwendet:

- Trenntransformator für das Oszilloskop: BLOCK Typ BR 802
- digitales Oszilloskop: RIGOL DS1104Z
- Frequenzgenerator: RIGOL DG1022
- Netzteil für die 12V- und -12V-Spannungsversorgung der Messschaltung und für die Gleichspannungen am Eingang der Messschaltung: Agilent E3631A

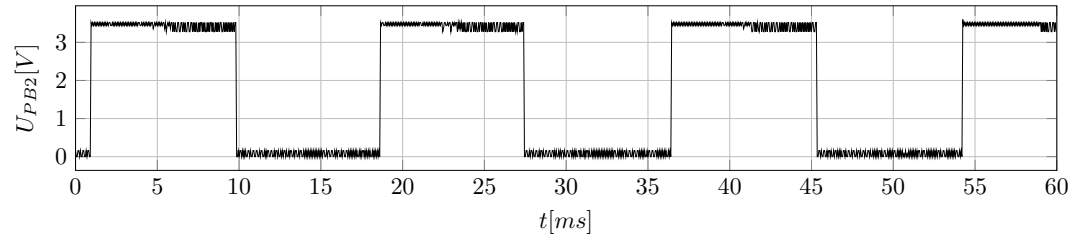
In Abb. 6.3 sind die Messergebnisse der Strommessschaltung mit OK_3 abgebildet. In Abb. 6.3a ist der Zusammenhang zwischen Ein- und Ausgangssignal zu sehen. Bis zu einer Eingangsspannung von $2V$ ist der Zusammenhang stark linear. Wird der Verlauf der Messwerte abschnittsweise durch Regressionsgeraden approximiert, kann eine Genauigkeit von bis zu $\Delta I = \pm 900 \mu A$ erreicht werden.



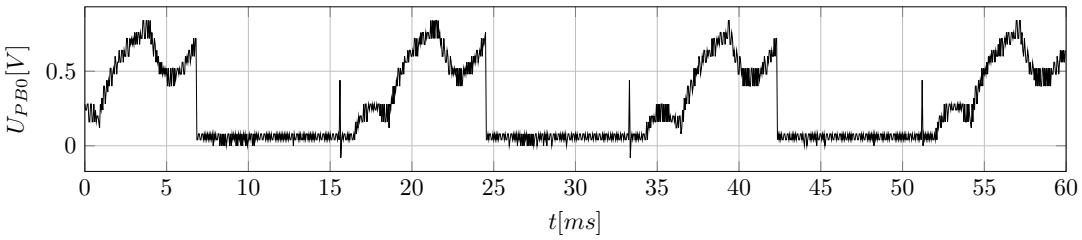
(a)



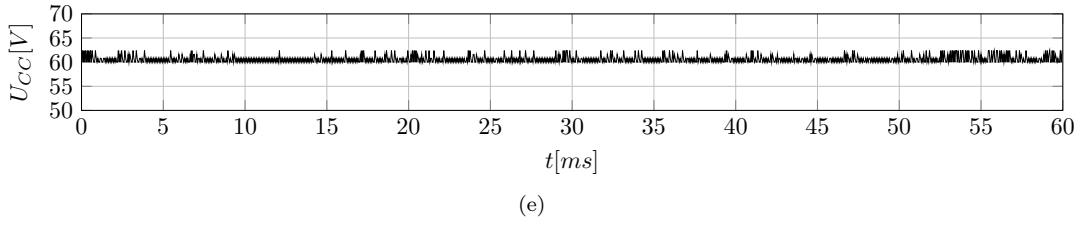
(b)



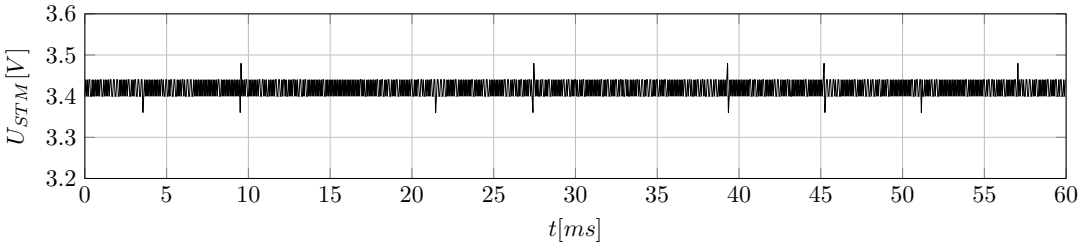
(c)



(d)



(e)



(f)

Abbildung 6.1: Messergebnisse für die Endstufen bei einer Rotation des Motors gegen den Uhrzeigersinn und einer Drehzahl von $2700 min^{-1}$ (im Leerlauf), Spannungen bezogen auf GND. (a) Spannung der Phase a, (b) Spannung der Phase c, (c) Spannung des GPIO-Pins PB2, (d) Spannung des ADC-Eingangs PB0, (e) Versorgungsspannung der Endstufen, (f) 3,3V-Versorgungsspannung

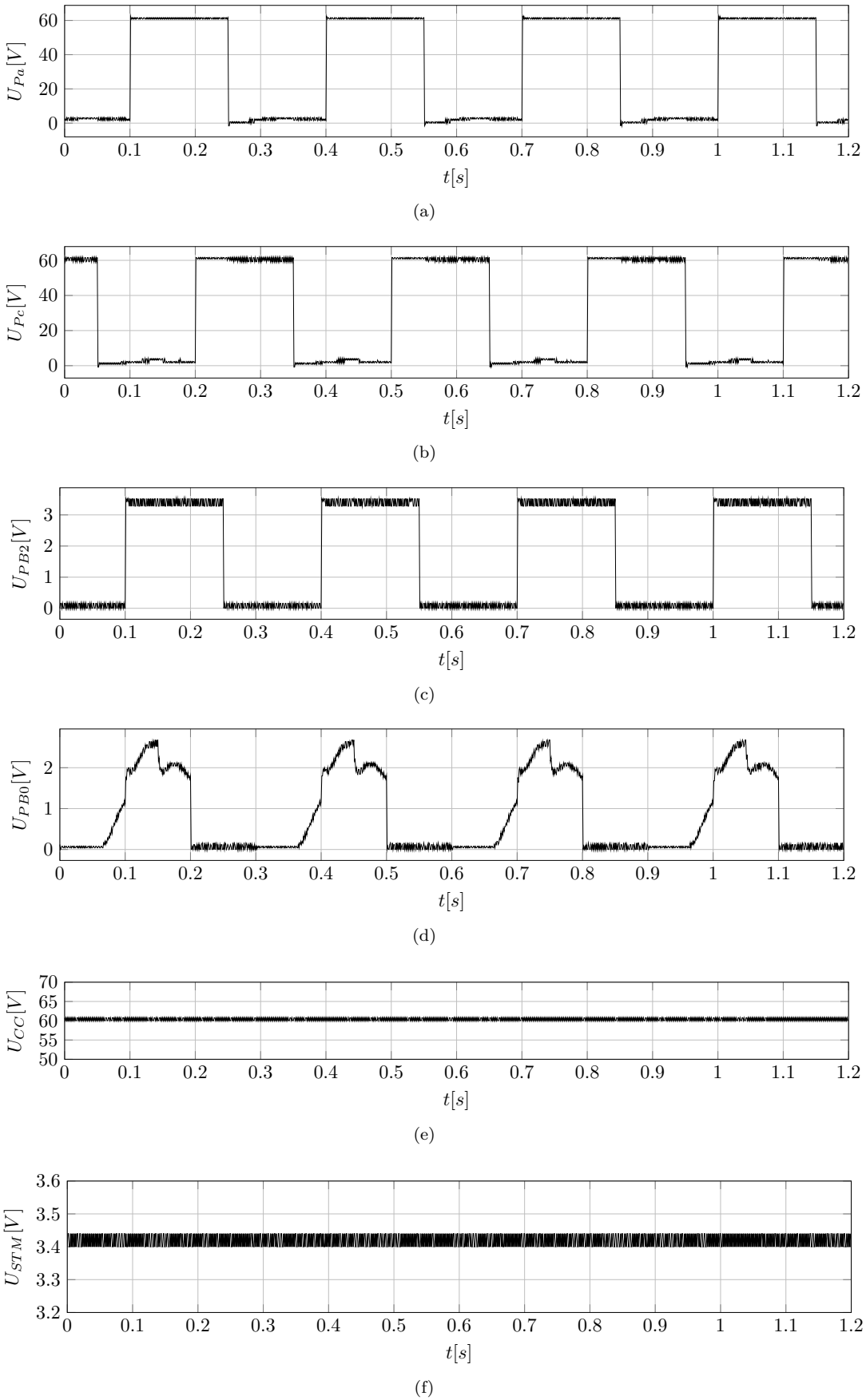


Abbildung 6.2: Messergebnisse für die Endstufen bei einer Rotation des Motors gegen den Uhrzeigersinn und einer Drehzahl von $210 min^{-1}$ (im Leerlauf), Spannungen bezogen auf GND. (a) Spannung der Phase a, (b) Spannung der Phase c, (c) Spannung des GPIO-Pins PB2, (d) Spannung des ADC-Eingangs PB0, (e) Versorgungsspannung der Endstufen, (f) 3,3V-Versorgungsspannung

Der Verlauf von U_{out} zu U_{in} für die anderen beiden Messschaltungen kann Abb. 8.6a bzw. Abb. 8.6b entnommen werden. In Abb. 6.3b bis 6.3d ist das Frequenzverhalten der Strommessschaltung zu sehen. Während für Frequenzen bis $10kHz$ fast keine Phasenverschiebung zu erkennen ist, zeigt das Ausgangssignal bei einer Frequenz von $40kHz$ einen klaren zeitlichen Versatz gegenüber dem Eingangssignal. Das Messen von Spannungen mit einer Frequenz von bis zu $10kHz$ ist also mit dieser Messschaltung möglich. Die Schaltung begrenzt auch die Ausgangsspannung auf einen Bereich von $0V \leq U_{out} \leq 3V$ (Abb. 6.3c), womit der zulässige Spannungsbereich für die ADC-Eingänge eingehalten wird. Abb. 6.3b bis 6.3d unterscheiden sich für die anderen Messschaltungen nur in einem nicht relevanten Maß.

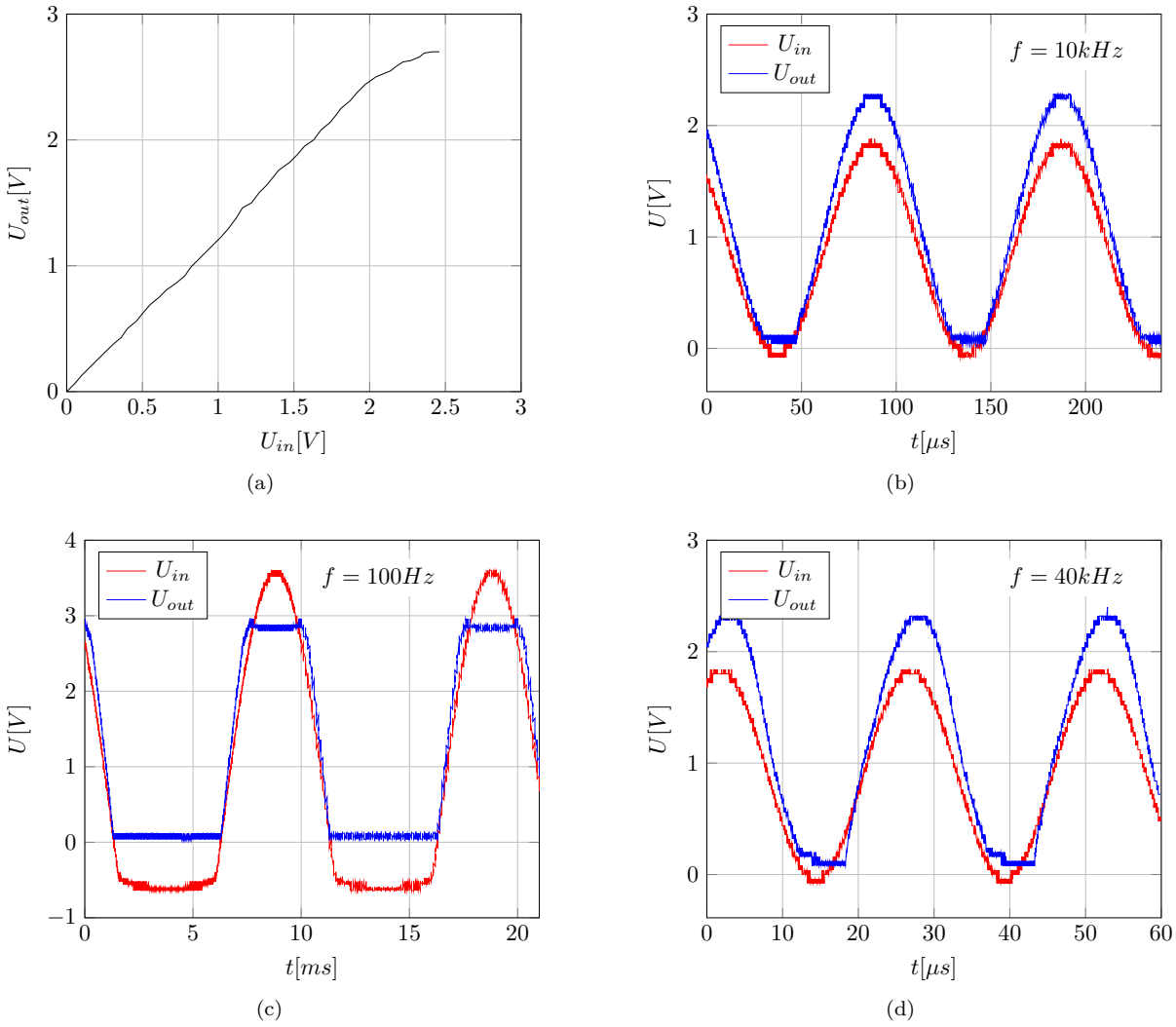


Abbildung 6.3: Messergebnisse für die Strommessschaltung mit OK_3 (Abb. 8.3). (a) Zusammenhang zwischen der Eingangs- und Ausgangsspannung, (b) Frequenzverhalten für $f = 10kHz$, (c) Frequenzverhalten für $f = 100Hz$, Frequenzverhalten für $f = 40kHz$. Resultierende Genauigkeit: $\Delta I = \pm 900\mu A$

6.3 Messergebnisse für die Phasenströme

Um den kompletten zeitlichen Verlauf der Phasenströme zu messen wurde ein Shunt in die entsprechende Leitung der Motorphase eingesetzt. Der Shunt ist identisch zum Widerstand R_{10} (siehe Anschnitt 4.2.5). Der gemessene Phasenstrom ist in Abb. 6.4 zu sehen. Die hohen Stromspitzen werden beim Abbau der Spannungsspitzen durch die Zener-Diode der oberen MOSFETs der Endstufen verursacht (siehe Abschnitt 4.2.1). Sie haben jedoch auf die Steuerung keinen negativen Einfluss. Der gewünschte sinusförmige Stromverlauf wird soweit durch den sehr einfachen Algorithmus aus Unterkapitel 5.2 angenähert, dass die Anforderungen aus Kapitel 1 erfüllt werden können. Die Phasenströme der anderen Phasen unterscheiden sich zu Abb. 6.4 nur durch eine Phasenverschiebung von $\pm 120°$.

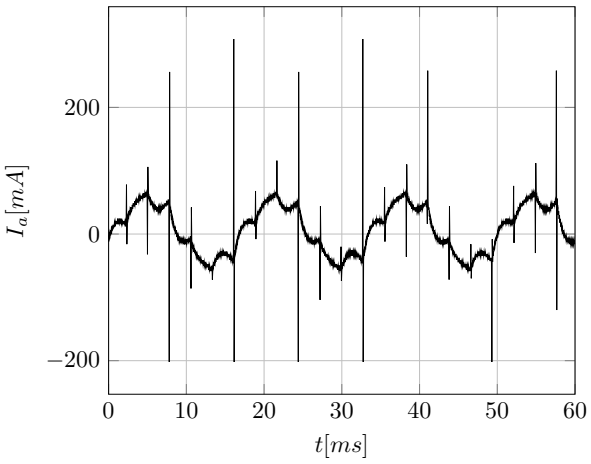
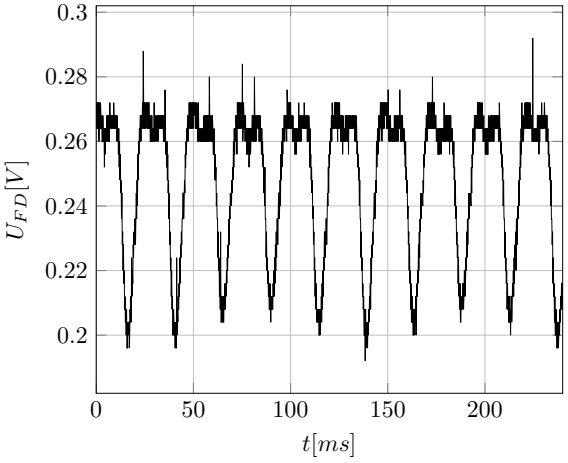


Abbildung 6.4: Messergebnis für den Phasenstrom $I_a(t)$ bei $f = 2700min^{-1}$

6.4 Messung der Drehzahl

Um die Drehzahl des Motors für die Messungen in Unterkapitel 6.2 zu ermitteln, wurde eine Fotodiode verwendet. Auf der Motorwelle wurde eine dunkle Markierung angebracht und über eine LED beleuchtet. Der Aufbau ist in Abb. 6.5a abgebildet. Die gemessene Spannung über der Fotodiode ist in Abb. 6.5b angegeben. Die Zeit zwischen zwei Minima von U_{FD} entspricht einer Umdrehung des Motors.

(a)



(b)

Abbildung 6.5: (a) Messaufbau für die Ermittlung der Drehzahl, (b) Messergebnis für die Spannung der Fotodiode bei $f = 2400min^{-1}$

6.5 Gesamtstromverbrauch der Platine

Für die Ermittlung des Gesamtstromverbrauchs wurde die interne Strommessung der Labornetzteile GwINSTEK GPS-4303 und Agilent E3631A verwendet. Zuerst wurde der Stromverbrauch bei Stillstand des Motors gemessen. Anschließend wurden die maximalen Ströme bei rotierendem Motor ermittelt. In Tabelle 6.1 sind die Messwerte angegeben.

	$f = 0$	$f \neq 0$
$I_{D13}[A]$	0	0, 19
$I_{D14}[A]$	0, 01	0, 033
$I_{D15}[A]$	0, 084	0, 118

Tabelle 6.1: Messwerte für den Gesamtstromverbrauch der Platine. I_{D13} bis I_{D15}: Ströme durch die Sperrdioden (siehe Abb. 8.1b), f: Drehzahl der Motors.

7 Fazit und Abschlussbemerkungen

7.1 Erfüllung der Anforderungen

Die erstellte Platine, mit dem implementierten Steueralgorithmus, erfüllt die Anforderungen bzgl. der Variabilität der Drehzahl und der Drehrichtung, sowie bzgl. der maximalen Drehzahl des Motors. Hier ist jedoch zu beachten, dass für die Verwendung eines anderen Motors, die Initialisierungswerte des Steueralgorithmus (Quellcode-Ausschnitt 5.6, Zeile 10-16) angepasst werden müssen . Die geplante Versorgungsspannung von 325V konnte, wie in Unterkapitel 4.5 beschrieben, nicht realisiert werden. Das Layout der Platine und die Bauteile sind jedoch für diese hohe Spannung ausgelegt worden. **Die Platine konnte jedoch für eine Versorgungsspannung von 325V nicht getestet werden!** Bei einem Betrieb der Platine mit einer Spannung höher als 60V sollten die Tests aus Kapitel 6 wiederholt werden. **Die Kondensatoren der Platine können hohe Restspannungen besitzen. Bevor Arbeiten an der Platine vorgenommen werden, muss immer die Restspannung geprüft werden!**

7.2 Das DTC-Regelverfahren

Das geplante Regelverfahren konnte aufgrund der zulangen Berechnungszeit nicht realisiert werden. Der Quellcode befindet sich in auskommentierter Form im Software-Projekt auf dem beiliegenden Datenträger. Um die Berechnungszeit zu reduzieren, können u.a. folgende Maßnahmen getroffen werden:

- Es kann ein Prozessor mit höherer Taktrate verwendet werden.
- Die Software-Ebenen können reduziert werden (z.B. können Registerzugriffe direkt durchgeführt werden, anstatt Bibliotheksfunktionen zu benutzen).
- Es kann ein FPGA verwendet werden. Dadurch kann eine hohe Parallelität des Algorithmus erreicht werden.

Des Weiteren sollten die Phasenströme direkt in den Motorphasen gemessen werden. Rückblickend ist die aktuelle Position der Shunts (R_{10}, R_{19} und R_{20} (Abb. 8.2)) nicht geeignet, um die Phasenströme bei allen Zuständen der Endstufen messen zu können. Es gibt Spannungs-RMZ, bei denen nur in einer Endstufe der untere MOSFET aktiv ist. Mit nur einem gemessenen Phasenstrom kann jedoch die Knotengleichung nicht gelöst werden. Daher muss ein Phasenstrom zusätzlich geschätzt werden. Ob ein Phasenstrom im dynamischen Betrieb des Motors ausreichend gut geschätzt werden kann ist fraglich. Daher ist es besser die Phasenströme in den Phasenleitungen zu messen.

8 Anhang

8.1 Schaltplan

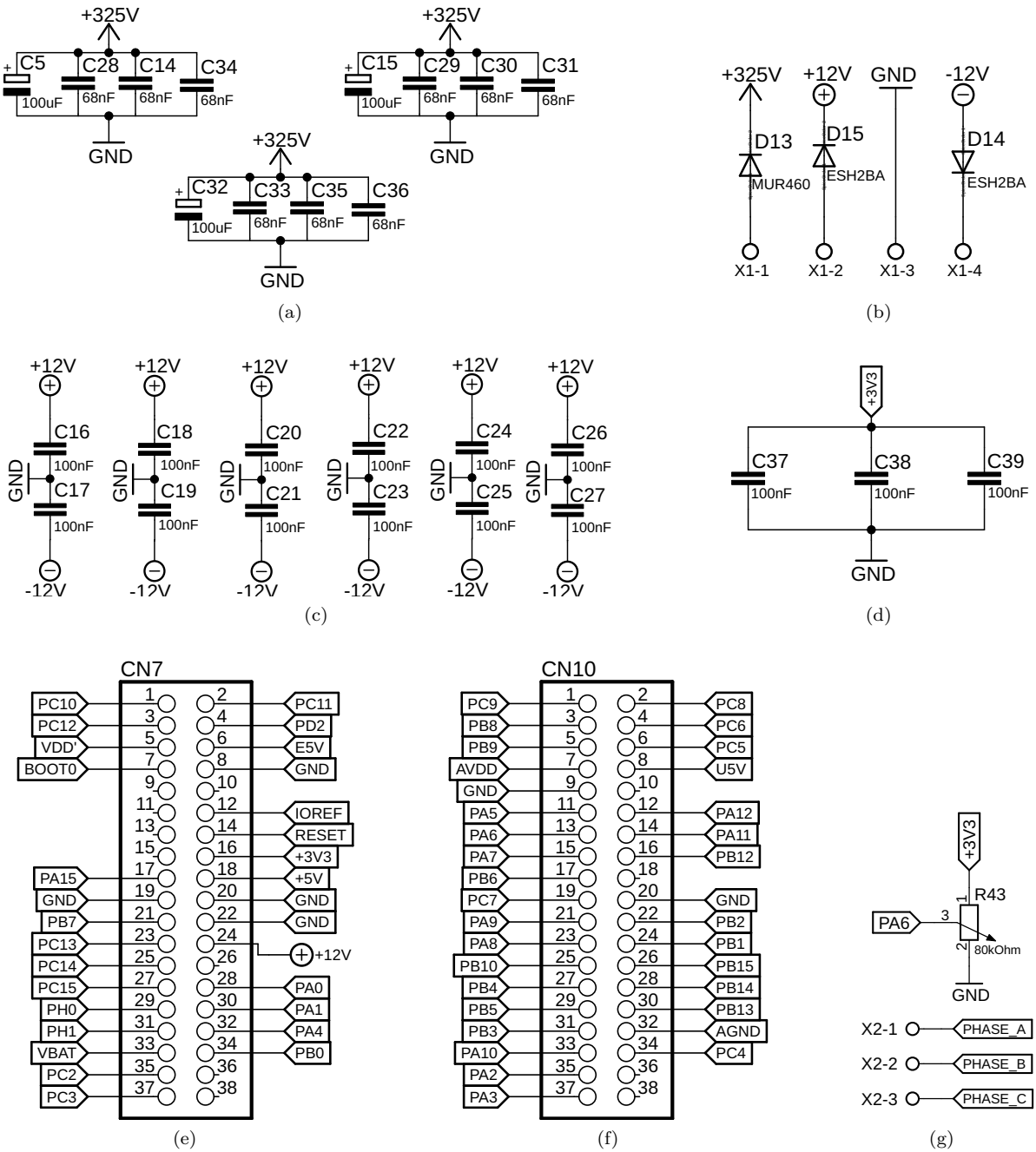


Abbildung 8.1: (a) Glättungskondensatoren für die Endstufen (Abb. 8.2), (b) Sperrdioden für die Spannungsversorgungen, (c) Entstörkondensatoren für OPV1 - OPV6 (Abb. 8.3), (d) Entstörkondensatoren für die 3,3V-Spannungsversorgung der Gate-Treiber IC1 - IC3 (Abb. 8.2), (e)/(f) Buchsenleisten für das STM32-Board (Beschriftung der einzelnen Buchsen ist identisch zur Beschriftung in [21, Tab. 25]), (g) Poti für das Einstellen der Motordrehzahl und Buchsenleiste für den Anschluss der Motorphasen

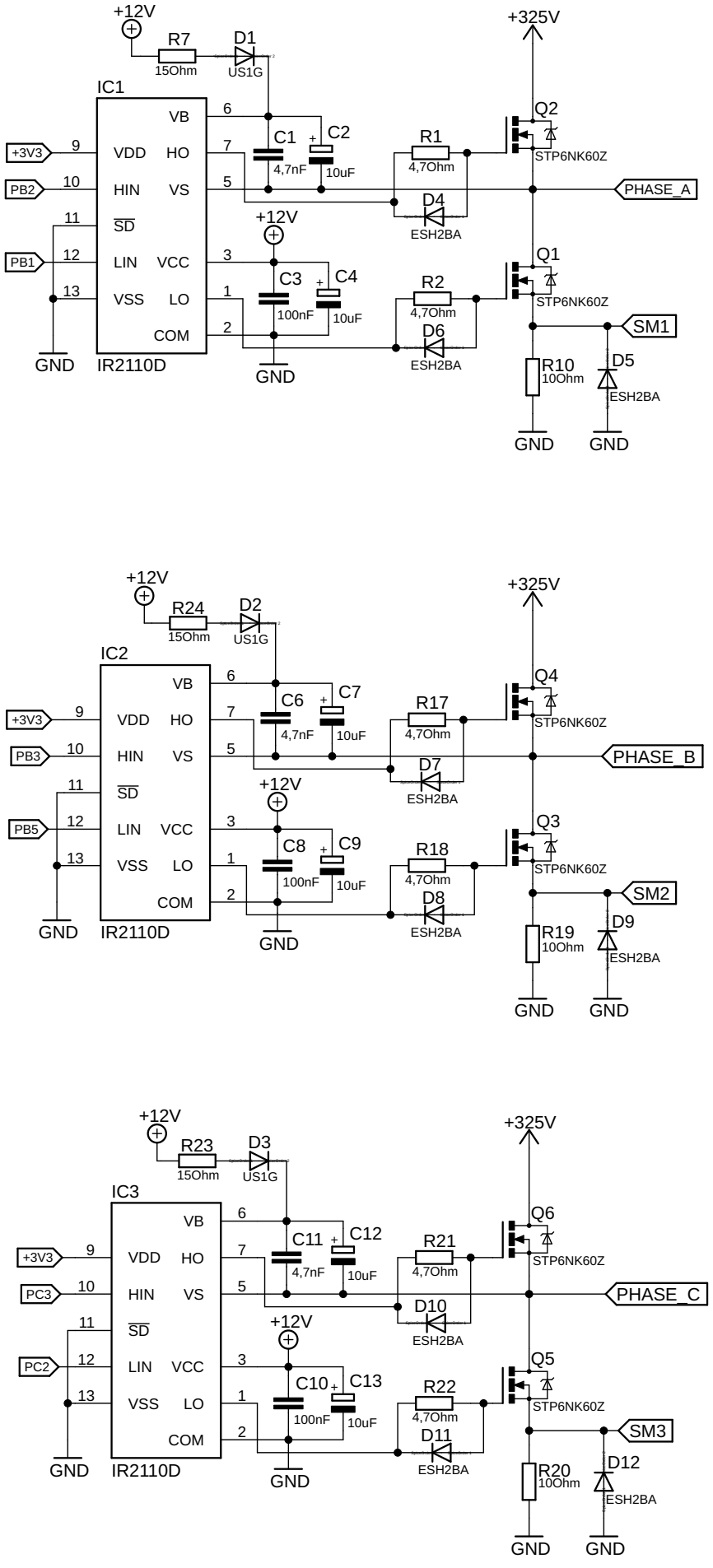


Abbildung 8.2: Endstufen des Umrichters

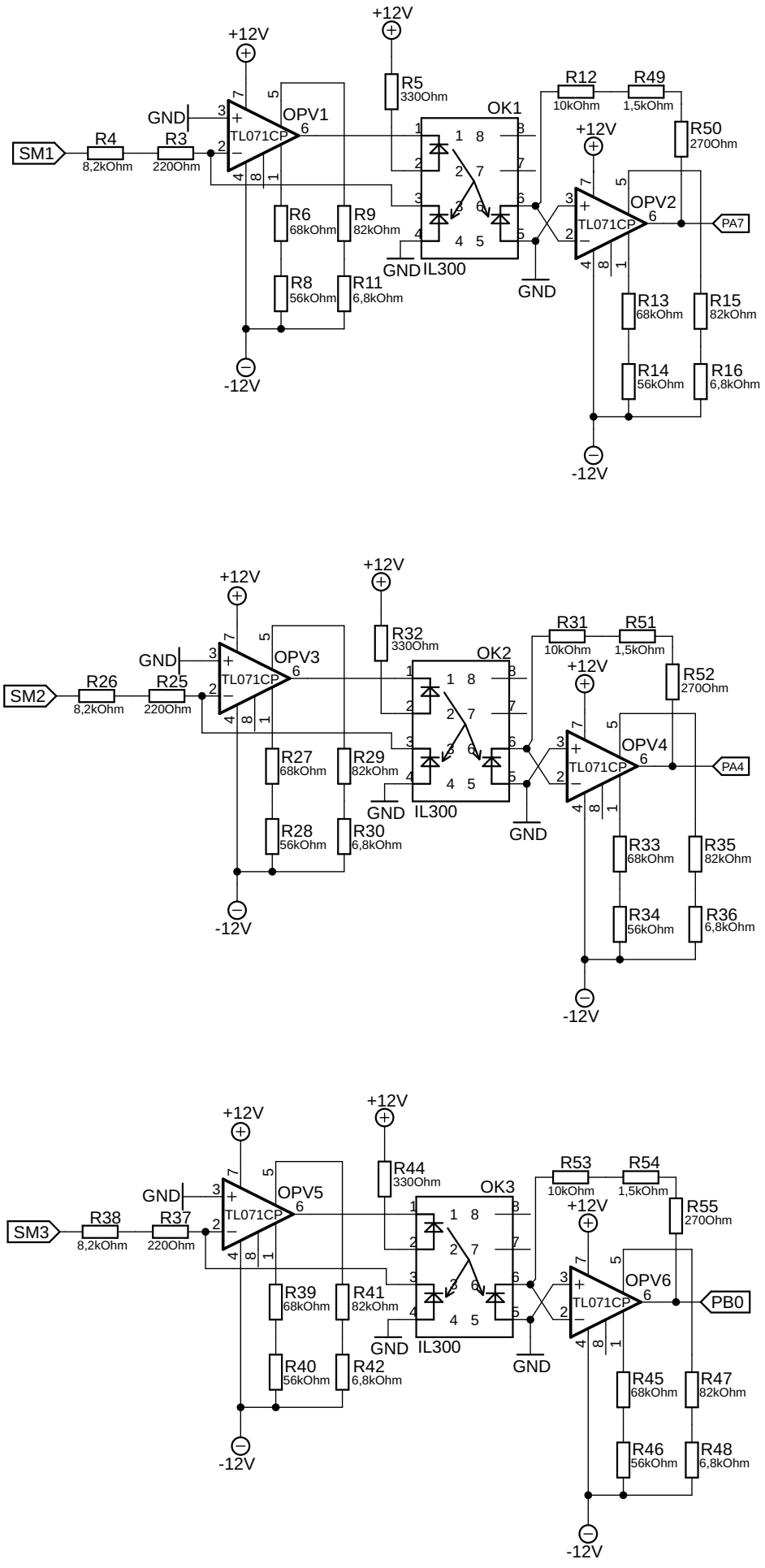
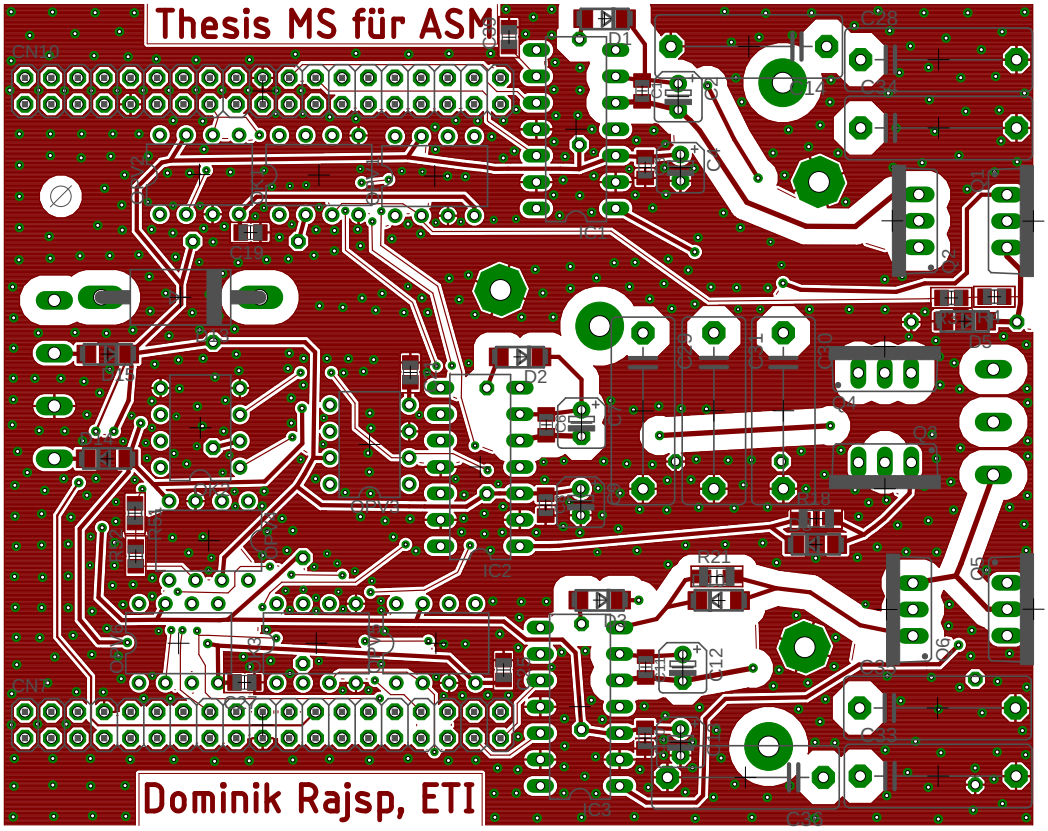
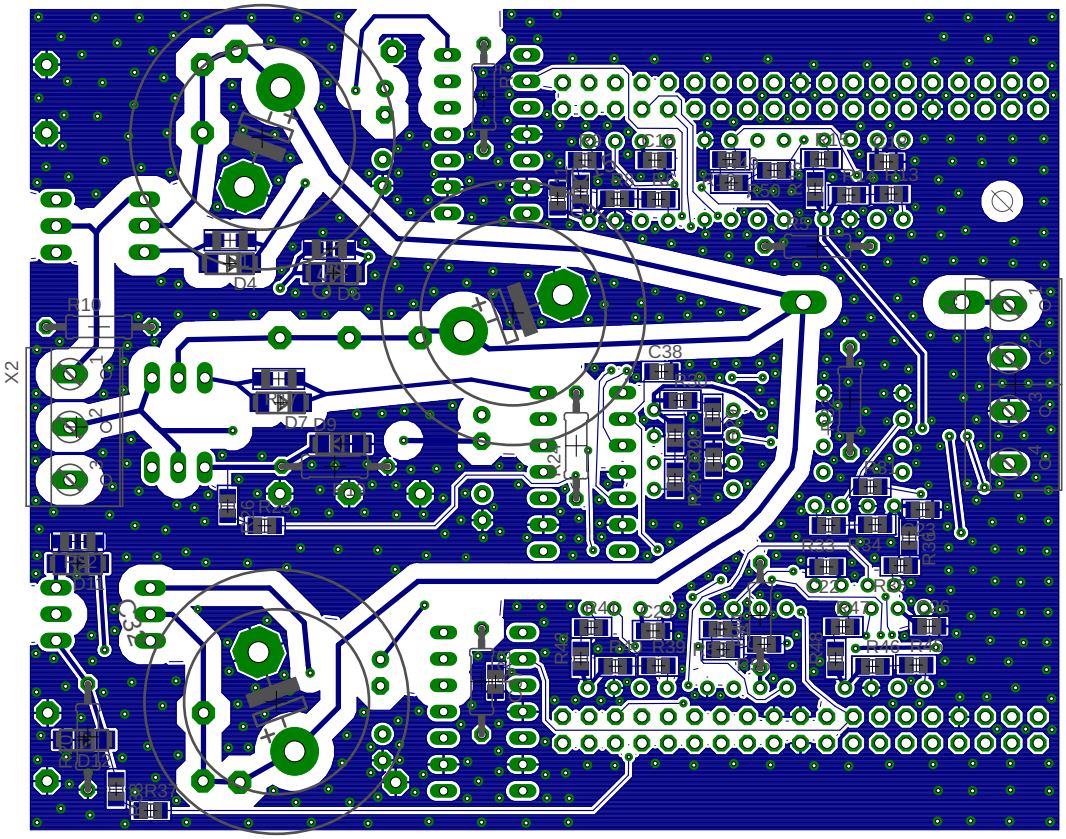


Abbildung 8.3: Strommessschaltungen

8.2 Layout der Platine

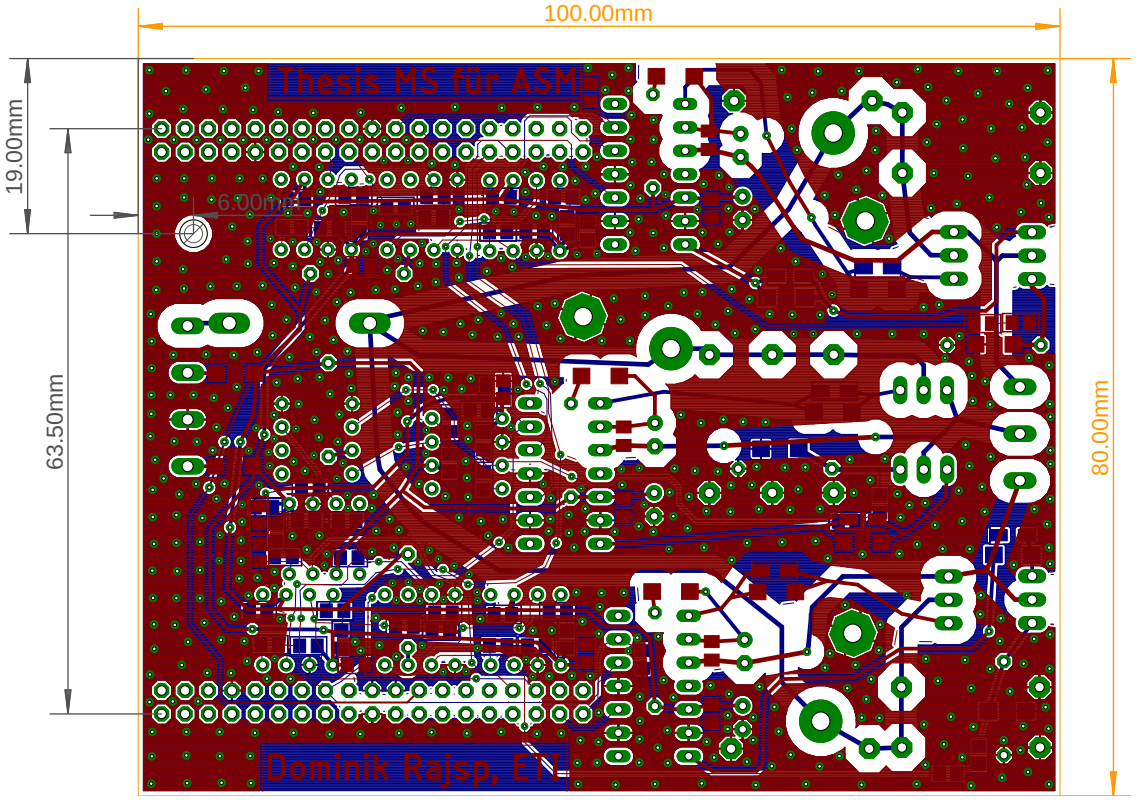


(a)

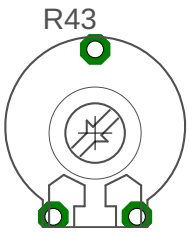


(b)

Abbildung 8.4: (a) Top-Layer der Platine, (b) Bottom-Layer der Platine



(a)



(b)

Abbildung 8.5: (a) gesamtes Layout der Umrichterplatine mit Bemaßung, Poti für das Einstellen der Motordrehzahl (Abb. 8.1g)

8.3 Literatur für die verwendeten Bauteile/Komponenten

(Schaltungs-)Komponenten	zugehörige Dokumente
Q1 - Q6	[19]
IC1 - IC3	[13], [7] [12], [11], [1]
D1 - D3	[18]
D4 - D12, D14, D15	[17]
D13	[16]
OPV1 - OPV 6	[5]
OK1 - OK3	[23], [22]
STM32-Board	[20], [21]
Motor	[6], [10], [2]

Tabelle 8.1: Literatur für die verwendeten Bauteile/Komponenten

8.4 Gesamter Quellcode für den Regelalgorithmus

```
#include "main.h"
#include "stm32f4xx_hal.h"
#include <stdbool.h>
#include <math.h>

ADC_HandleTypeDef hadc1;
DMA_HandleTypeDef hdma_adc1;
uint32_t adcData[4]; //ADC-Daten der 4 ADC-Kanäle,
{PA7.data, PA4.data, PB0.data,PA6.data}
int k; //enspricht Verzögerungszeit zwischen den
einzelnen Spannungs-RMZ

void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_ADC1_Init(void);

int cw; //cw=1-> Rotation im Uhrzeigersinn,
cw=0-> Rotation gegen den Uhrzeigersinn. cw=2-> stop
int cwBegin; //ADC-Wert ab dem der CW-Bereich beginnt
int ccwBegin;//ADC-Wert ab dem der CCW-Bereich beginnt
int cwOffset; //Offset für die Gerade im CW-Bereich
int ccwOffset;//Offset für die Gerade im CCW-Bereich
int mcw; //Steigung der Gerade für den CW-Bereich
int mccw; //Steigung der Gerade für den CW-Bereich
int hysK; //Hysteresebreite für k

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
        int x=0;

        if(adcData[3]>ccwBegin)
        {
                cw=0;
                x =mccw*(adcData[3]-ccwBegin)+ccwOffset;
        }
        else
        {
                if(adcData[3]<cwBegin)
                {
                        cw=1;
                        x =mcw*adcData[3]+cwOffset;
                }
                else
                {
                        cw=2;
                }
        }

        if((k-x<-hysK)||(k-x>hysK)) k=x;
}


//Definition der Spannungs-RMZ
void U11(void)
{
        //MOSFETs abschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_RESET); //MOSFET Q1
```

```
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_3,GPIO_PIN_RESET); //MOSFET Q4
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_3,GPIO_PIN_RESET); //MOSFET Q6

        //MOSFETs einschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_2,GPIO_PIN_SET); //MOSFET Q2
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_SET); //MOSFET Q3
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_2,GPIO_PIN_SET); //MOSFET Q5
}

void U12(void)
{
        //MOSFETs abschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_RESET); //MOSFET Q1
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_RESET); //MOSFET Q3
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_3,GPIO_PIN_RESET); //MOSFET Q6

        //MOSFETs einschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_2,GPIO_PIN_SET); //MOSFET Q2
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_3,GPIO_PIN_SET); //MOSFET Q4
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_2,GPIO_PIN_SET); //MOSFET Q5
}

void U13(void)
{
        //MOSFETs abschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_2,GPIO_PIN_RESET); //MOSFET Q2
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_RESET); //MOSFET Q3
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_3,GPIO_PIN_RESET); //MOSFET Q6

        //MOSFETs einschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_SET); //MOSFET Q1
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_3,GPIO_PIN_SET); //MOSFET Q4
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_2,GPIO_PIN_SET); //MOSFET Q5
}

void U14(void)
{
        //MOSFETs abschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_2,GPIO_PIN_RESET); //MOSFET Q2
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_RESET); //MOSFET Q3
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_2,GPIO_PIN_RESET); //MOSFET Q5

        //MOSFETs einschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_SET); //MOSFET Q1
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_3,GPIO_PIN_SET); //MOSFET Q4
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_3,GPIO_PIN_SET); //MOSFET Q6
}

void U15(void)
{
        //MOSFETs abschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_2,GPIO_PIN_RESET); //MOSFET Q2
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_3,GPIO_PIN_RESET); //MOSFET Q4
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_2,GPIO_PIN_RESET); //MOSFET Q5

        //MOSFETs einschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_SET); //MOSFET Q1
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_SET); //MOSFET Q3
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_3,GPIO_PIN_SET); //MOSFET Q6
}
```

```
void U16(void)
{
        //MOSFETs abschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_RESET); //MOSFET Q1
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_3,GPIO_PIN_RESET); //MOSFET Q4
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_2,GPIO_PIN_RESET); //MOSFET Q5

        //MOSFETs einschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_2,GPIO_PIN_SET); //MOSFET Q2
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_SET); //MOSFET Q3
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_3,GPIO_PIN_SET); //MOSFET Q6
}

void U17(void)
{
        //MOSFETs abschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_RESET); //MOSFET Q1
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_RESET); //MOSFET Q3
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_2,GPIO_PIN_RESET); //MOSFET Q5

        //MOSFETs einschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_2,GPIO_PIN_SET); //MOSFET Q2
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_3,GPIO_PIN_SET); //MOSFET Q4
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_3,GPIO_PIN_SET); //MOSFET Q6
}

void U18(void)
{
        //MOSFETs abschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_2,GPIO_PIN_RESET); //MOSFET Q2
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_3,GPIO_PIN_RESET); //MOSFET Q4
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_3,GPIO_PIN_RESET); //MOSFET Q6

        //MOSFETs einschalten
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_1,GPIO_PIN_SET); //MOSFET Q1
        HAL_GPIO_WritePin(GPIOB,GPIO_PIN_5,GPIO_PIN_SET); //MOSFET Q3
        HAL_GPIO_WritePin(GPIOC,GPIO_PIN_2,GPIO_PIN_SET); //MOSFET Q5
}

typedef void (*spRMZ) (void); //Daten-Typ für Funktionszeiger

spRMZ swTable[2][6]={{U11,U16,U15,U14,U13,U12},
//Schalttabelle für Rotation gegen den Uhrzeigersinn
                     {U11,U12,U13,U14,U15,U16}};
//Schalttabelle für Rotation im Uhrzeigersinn


int main(void)
{
  HAL_Init();

  SystemClock_Config();

  MX_GPIO_Init();
  MX_DMA_Init();
  MX_ADC1_Init();

  ccwBegin = 2150;
  cwBegin = 1950;
  mccw = -485;
  mcw = 485;
```

```
  ccwOffset = 1000000;
  cwOffset = 55000;
  hysK =50;


  HAL_ADC_Start_DMA(&hadc1,adcData,4); //ADC in DMA-Mode starten

  int i=0;

  while (1)
  {
        if(cw<2)
        {
              swTable[cw][i]();
              if(i<5) i++;
              else i=0;
              for(int j=0; j<k;j++);
        }
        else
        {
              U18();
        }
  }

}



/** System Clock Configuration
*/
void SystemClock_Config(void)
{

  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;

    /**Configure the main internal regulator output voltage
    */
  __HAL_RCC_PWR_CLK_ENABLE();

  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

    /**Initializes the CPU, AHB and APB busses clocks
    */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = 16;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 16;
  RCC_OscInitStruct.PLL.PLLN = 200;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Initializes the CPU, AHB and APB busses clocks
    */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
```

```
                                    |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
  FLASH_LATENCY_3) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Configure the Systick interrupt time
    */
  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

    /**Configure the Systick
    */
  HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

  /* SysTick_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}

/* ADC1 init function */
static void MX_ADC1_Init(void)
{

  ADC_ChannelConfTypeDef sConfig;

  /**Configure the global features of the ADC
   (Clock, Resolution, Data Alignment
    and number of conversion)
  */
  hadc1.Instance = ADC1;
  hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
  hadc1.Init.Resolution = ADC_RESOLUTION_12B;
  hadc1.Init.ScanConvMode = ENABLE;
  hadc1.Init.ContinuousConvMode = ENABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 4;
  hadc1.Init.DMAContinuousRequests = ENABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  /**Configure for the selected ADC regular channel its
     corresponding rank in the sequencer and
     its sample time.
  */
  sConfig.Channel = ADC_CHANNEL_7;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_480CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
```

```
  }                                                                    303
                                                                       304
  /**Configure for the selected ADC regular channel                    305
    its corresponding rank in the sequencer and                        306
    its sample time.                                                   307
  */                                                                   308
  sConfig.Channel = ADC_CHANNEL_4;                                     309
  sConfig.Rank = 2;                                                    310
        sConfig.SamplingTime = ADC_SAMPLETIME_480CYCLES;               311
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)               312
  {                                                                    313
    _Error_Handler(__FILE__, __LINE__);                                314
  }                                                                    315
                                                                       316
  /**Configure for the selected ADC regular channel                    317
    its corresponding rank in the sequencer and                        318
    its sample time.                                                   319
  */                                                                   320
  sConfig.Channel = ADC_CHANNEL_8;                                     321
  sConfig.Rank = 3;                                                    322
        sConfig.SamplingTime = ADC_SAMPLETIME_480CYCLES;               323
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)               324
  {                                                                    325
    _Error_Handler(__FILE__, __LINE__);                                326
  }                                                                    327
                                                                       328
  /**Configure for the selected ADC regular channel                    329
    its corresponding rank in the sequencer and                        330
    its sample time.                                                   331
  */                                                                   332
  sConfig.Channel = ADC_CHANNEL_6;                                     333
  sConfig.Rank = 4;                                                    334
        sConfig.SamplingTime = ADC_SAMPLETIME_480CYCLES;               335
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)               336
  {                                                                    337
    _Error_Handler(__FILE__, __LINE__);                                338
  }                                                                    339
                                                                       340
}                                                                      341
                                                                       342
/**                                                                    343
  * Enable DMA controller clock                                        344
  */                                                                   345
static void MX_DMA_Init(void)                                          346
{                                                                      347
  /* DMA controller clock enable */                                    348
  __HAL_RCC_DMA2_CLK_ENABLE();                                         349
                                                                       350
  /* DMA interrupt init */                                             351
  /* DMA2_Stream0_IRQn interrupt configuration */                      352
  HAL_NVIC_SetPriority(DMA2_Stream0_IRQn, 0, 0);                       353
  HAL_NVIC_EnableIRQ(DMA2_Stream0_IRQn);                               354
                                                                       355
}                                                                      356
                                                                       357
/** Configure pins as                                                 358
        * Analog                                                       359
        * Input                                                        360
        * Output                                                       361
        * EVENT_OUT                                                   362
        * EXTI                                                        363
```

```
*/                                                                              364
static void MX_GPIO_Init(void)                                                  365
{                                                                               366
                                                                                367
  GPIO_InitTypeDef GPIO_InitStruct;                                             368
                                                                                369
  /* GPIO Ports Clock Enable */                                                 370
  __HAL_RCC_GPIOC_CLK_ENABLE();                                                 371
  __HAL_RCC_GPIOA_CLK_ENABLE();                                                 372
  __HAL_RCC_GPIOB_CLK_ENABLE();                                                 373
                                                                                374
  /*Configure GPIO pin Output Level */                                        375
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_2|GPIO_PIN_3, GPIO_PIN_RESET);              376
                                                                                377
  /*Configure GPIO pin Output Level */                                        378
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|                    379
  GPIO_PIN_5, GPIO_PIN_RESET);                                                  380
                                                                                381
  /*Configure GPIO pins : PC2 PC3 */                                          382
  GPIO_InitStruct.Pin = GPIO_PIN_2|GPIO_PIN_3;                                  383
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;                                   384
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;                                         385
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;                            386
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);                                       387
                                                                                388
  /*Configure GPIO pins : PB1 PB2 PB3 PB5 */                                  389
  GPIO_InitStruct.Pin = GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|GPIO_PIN_5;            390
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;                                   391
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;                                         392
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;                            393
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);                                       394
                                                                                395
}                                                                               396
```

8.5 Zusätzliche Messergebnisse

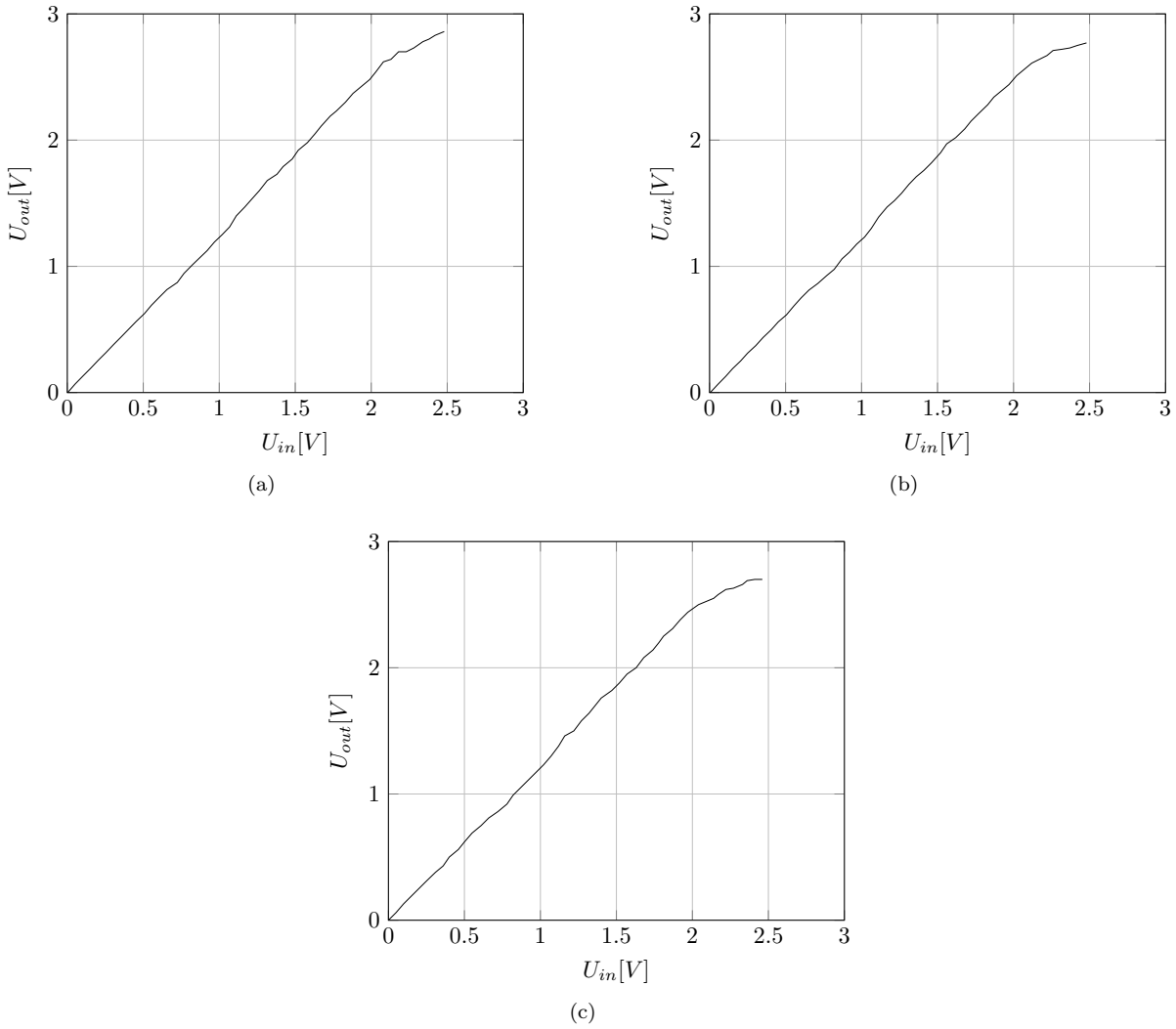


Abbildung 8.6: Messergebnisse für die Strommessschaltungen. (a) U_{out}/U_{in} für die Messschaltung mit OK_1, (b) U_{out}/U_{in} für die Messschaltung mit OK_2, U_{out}/U_{in} für die Messschaltung mit OK_3 (siehe Abb. 8.3).

Tabelle 8.2: Tabelle mit den Messwerten zu Abb. 8.6

Strommessschaltung mit OK_1		Strommessschaltung mit OK_2		Strommessschaltung mit OK_3	
$U_{in}[V]$	$U_{out}[V]$	$U_{in}[V]$	$U_{out}[V]$	$U_{in}[V]$	$U_{out}[V]$
0,004	0,002	0,006	0,002	0,000	0,002
0,052	0,068	0,054	0,064	0,054	0,064
0,104	0,132	0,106	0,128	0,104	0,130
0,152	0,190	0,152	0,190	0,154	0,188
0,204	0,256	0,204	0,250	0,202	0,250
0,254	0,316	0,250	0,312	0,254	0,314
0,306	0,382	0,304	0,372	0,306	0,376
0,354	0,440	0,352	0,436	0,356	0,434
0,404	0,502	0,406	0,498	0,404	0,498
0,454	0,564	0,452	0,560	0,456	0,558
0,508	0,626	0,506	0,616	0,504	0,620

Fortsetzung auf der nächsten Seite

Fortsetzung

Strommessschaltung mit OK_1		Strommessschaltung mit OK_2		Strommessschaltung mit OK_3	
$U_{in}[V]$	$U_{out}[V]$	$U_{in}[V]$	$U_{out}[V]$	$U_{in}[V]$	$U_{out}[V]$
0,554	0,692	0,556	0,688	0,554	0,688
0,606	0,756	0,604	0,752	0,606	0,752
0,656	0,816	0,654	0,812	0,656	0,808
0,724	0,872	0,712	0,864	0,720	0,864
0,768	0,944	0,772	0,928	0,776	0,924
0,820	1,008	0,820	0,976	0,816	0,992
0,868	1,064	0,872	1,060	0,868	1,048
0,920	1,124	0,920	1,112	0,920	1,112
0,968	1,192	0,968	1,176	0,968	1,172
1,020	1,252	1,020	1,232	1,020	1,232
1,068	1,312	1,064	1,300	1,068	1,300
1,112	1,400	1,112	1,390	1,116	1,380
1,168	1,470	1,168	1,470	1,164	1,460
1,212	1,530	1,216	1,520	1,220	1,500
1,264	1,600	1,264	1,580	1,268	1,580
1,316	1,680	1,312	1,650	1,316	1,640
1,380	1,730	1,360	1,710	1,364	1,700
1,420	1,790	1,410	1,760	1,400	1,760
1,480	1,850	1,460	1,820	1,470	1,820
1,520	1,920	1,520	1,900	1,520	1,880
1,580	1,980	1,560	1,970	1,570	1,950
1,630	2,050	1,620	2,020	1,630	2,000
1,670	2,110	1,680	2,090	1,680	2,080
1,730	2,190	1,720	2,150	1,740	2,140
1,770	2,230	1,770	2,210	1,780	2,200
1,830	2,300	1,830	2,280	1,810	2,250
1,880	2,370	1,870	2,340	1,870	2,310
1,930	2,420	1,920	2,390	1,920	2,380
1,990	2,480	1,970	2,440	1,970	2,440
2,030	2,540	2,020	2,510	2,040	2,500
2,080	2,620	2,080	2,570	2,060	2,510
2,130	2,640	2,120	2,610	2,140	2,550
2,180	2,700	2,170	2,640	2,170	2,580
2,230	2,700	2,220	2,670	2,220	2,620
2,280	2,730	2,260	2,710	2,270	2,630
2,340	2,780	2,320	2,720	2,330	2,660
2,380	2,800	2,370	2,730	2,360	2,690
2,420	2,830	2,420	2,750	2,410	2,700
2,480	2,860	2,480	2,770	2,460	2,700

Literaturverzeichnis

[1] M. Grasso A. Merello A. Rugginenti. *Using Monolithic High Voltage Gate Drivers*. doc. no.: DT04-04. Application Note. International Rectifier.

[2] Amann. *KD/DR62.0*. Technische Zeichnung. Dunktermotoren GmbH. 18. Nov. 2015.

[3] Andreas Binder. *Elektrische Maschinen und Antriebe, Gundlagen und Betriebsverhalten*. 1. Auflage. ISBN 978-3-540-718505-5 (eBook). Springer, 2012.

[4] B. Sarvesh H. Sudheer S.F. Kodad. „Improvements in direct torque control of induction motor for wide range of speed operation using fuzzy logic“. In: *Journal of Electrical Systems and Information Technology* 155 (2017).

[5] Texas Instruments. *TL07x Low-Noise JFET-Input Operational Amplifiers*. Version rev. L. Datasheet. 1. Feb. 2014.

[6] Korthauer. *DR62.0x60-2*. Datenblatt. Dunktermotoren GmbH. 16. Juli 2013.

[7] Satyavrat Laud. *Application Note AN-1092. Understanding HVIC Datasheet Specifications*. International Rectifier.

[8] mikrocontroller.net. *Leiterbahnabstände. Mindestkriechstrecken bei Leiterplatten*. URL: `https://www.mikrocontroller.net/articles/Leiterbahnabst%C3%A4nde` (besucht am 14. 06. 2018).

[9] Wolfgang Notling. *Grundkurs Theoretische Physik 3, Elektrodynamik*. 10. Auflage. ISBN 978-3-642-37905-5 (eBook). Springer Spektrum, 2013.

[10] *Original-Montageanleitung KD/DR*. Installationsanleitung. Dunktermotoren GmbH. 30. Nov. 2017.

[11] International Rectifier. *Application Note AN-937. Gate Drive Characteristics and Requirements for HEXFET Power MOSFETs*.

[12] International Rectifier. *Application Note AN-978. HV Floating MOS-Gate Driver ICs*. Version rev. D. 23. März 2007.

[13] International Rectifier. *IR2110(S)/IR2113(S). HIGH AND LOW SIDE DRIVER*. doc. no.: PD60147. Version rev. T. Datasheet. 23. März 2004.

[14] Dierk Schröder. *Elektrische Antriebe - Grundlagen*. 6. Auflage. ISBN 978-3-662-55448-7 (eBook). Springer Vieweg, 2017.

[15] Dierk Schröder. *Elektrische Antriebe - Regelung von Antriebssystemen*. 4. Auflage. ISBN 978-3-642-30096-7 (eBook). Springer Vieweg, 2015.

[16] ON Semiconductor. *MUR405, MUR410, MUR415, MUR420, MUR440, MUR460. SWITCHMODE Power Rectifiers*. doc. id: MUR420/D. Version rev. 11. Datasheet. 1. Juli 2006.

[17] TAIWAN Semiconductor. *ESH2BA - ESH2DA. Surface Mount Super Fast Rectifiers*. Version version A11. Datasheet.

[18] TAIWAN Semiconductor. *US1A - US1M. High Efficient Surface Mount Rectifiers*. Version version G11. Datasheet.

[19] ST. *STB6NK60Z - STB6NK60Z-1/STP6NK60Z - STP6NK60ZFP. N-CHANNEL Zener-Protected SuperMESH MOSFET*. Version rev. 3. Datasheet. 1. Sep. 2005.

[20] ST. *STM32F411xC STM32F411xE*. doc. id: 026289. Version rev. 7. Datasheet. 1. Dez. 2017.

[21] ST. *UM1724. User manual STM32 Nucleo boards*. doc. id: 025833. Version rev. 7. 1. Jan. 2015.

[22] vishay. *ApplicationNote 50*. doc. no.: 83708. Version rev. 1.4. 8. Juni 2007.

[23] vishay. *IL300*. doc. no.: 83622. Version rev. 1.5. Datasheet. 24. März 2005.

Abkürzungsverzeichnis

ASM	3-Phasen-Asynchronmotor mit Kurzschlussläufer
RMZ	Raumzeiger
DTC	Direct Torque Control
PAP	Programmablaufplan
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
OPV	Operationsverstärker
IC	Integrated Circuit
Elko	Elektrolytkondensator
ADC	Analog-to-Digital Converter
LED	Light-Emitting Diode
DMA	Direct Memory Access
GPIO	General-Purpose Input/Output
FPGA	Field Programmable Gate Array